MARKOV RANDOM FIELD MODEL BASED TEXT SEGMENTATION AND IMAGE POST PROCESSING OF COMPLEX SCANNED DOCUMENTS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Eri Haneda

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2011

Purdue University

West Lafayette, Indiana

For my parents, Haruko and Hiromasa.

ACKNOWLEDGMENTS

First, I would like to express my greatest appreciation to my major advisor, Professor Charles A. Bouman, who has given me countless precious advice, encouragement, consideration, support, and discipline to train me. His creativity and enthusiasm toward research and education has been a great inspiration. The time I spent in this lab is so memorable. I would also like to thank Professor Jan P. Allebach for his great support and consideration. He provided both research and teaching opportunities that allowed me to improve my professional skills. I would also like to thank Professor George Chiu for his warm encouragement, and for setting up comfortable research environment. I would also like to thank Professor Peter C. Doerschuk, who introduced me to Professor Bouman. I could not accomplish my Ph.D. without his support. I would also like to thank all my colleagues including Guotong Feng, Mustafa Kamasak, Hasib Siddiqui, Maribel Figuera, William Wong, Burak Bitlis, Byungseok Min, DongOk Kim, Leonardo Bachega, Dalton Lunga, Haitao Xue, and Yandong Guo. Finally, I would like to express special thanks to Jordan Kisner, one of my best supporters, who gave me tremendous encouragement.

TABLE OF CONTENTS

Page

LIST OF TABLES vi

LIST OF FIGURES vii

SYMBOLS xii

ABBREVIATIONS xiv

ABSTRACT xv

1 SEGMENTATION FOR MRC DOCUMENT COMPRESSION USING A MRF MODEL 1

1.1 Introduction 1

1.2 Cost Optimized Segmentation (COS) 7

1.2.1 Blockwise segmentation 7

1.2.2 Global segmentation 8

1.3 Connected Component Classification (CCC) 14

1.3.1 Markov random field model 14

1.3.2 Connected component classification (CCC) algorithm 16

1.3.3 Statistical model 19

1.3.4 Inference 22

1.3.5 Parameter estimation 25

1.4 Multiscale-COS/CCC Segmentation Scheme 26

1.5 MRC Encoding 30

1.5.1 Subsampling 31

1.5.2 Data filling 33

1.6 Results 36

1.6.1 Preprocessing 37

1.6.2 Segmentation accuracy and bitrate 39

Page

1.6.3 Computation time 45
1.6.4 Qualitative results 46
1.6.5 Prior model evaluation 47
1.7 Summary 62
2 AUTOMATIC CONTRAST ENHANCEMENT SCHEME FOR A DIGITAL SCANNED IMAGE 63
2.1 Introduction 63
2.2 Auto Cropping 66
2.3 Paper-white Estimation 71
2.4 Black Colorant Estimation 74
2.5 Contrast Stretch 76
2.6 Show-through Blocking 76
2.7 Results 77
2.7.1 Paper-white estimation 77
2.7.2 Qualitative results 78
2.7.3 Show-through improvements 82
2.8 Summary 85
LIST OF REFERENCES 86
A FEATURE VECTOR FOR CCC 91

LIST OF TABLES

Table Page

1.1 Parameter settings for the COS algorithm in multiscale-COS/CCC. . . 40

1.2 Segmentation accuracy comparison between our algorithms (multiscale-COS/CCC, COS/CCC, and COS), thresholding algorithms (Otsu and Tsai), an MRF-based algorithm (multiscale-COS/CCC/Zheng), and two commercial MRC document compression packages (DjVu and LuraDocument). Missed component error, p_{MC}, the corresponding missed pixel error, p_{MP}, false component error, p_{FC}, and the corresponding false pixel error, p_{FP}, are calculated for EPSON, HP, and Samsung scanner output. 42

1.3 Comparison of bitrate between multiscale-COS/CCC, multiscale-COS/CCC/Zheng, DjVu, LuraDocument, Otsu/CCC, and Tsai/CCC for JBIG2 compressed binary mask layer for images scanned on EPSON, HP, and Samsung scanners. 45

1.4 Computation time of multiscale-COS/CCC algorithms with 3 layers, 2 layers, COS-CCC, COS, and Multiscale-COS/CCC/Zheng. 46

2.1 Parameter settings for automatic contrast enhancement 77

2.2 Paper-white estimation error for $Thres_{PW} = 10$ 78

2.3 Paper-white estimation error for $Thres_{PW} = 14$ 79

2.4 Paper-white estimation error for $Thres_{PW} = 17$ 79

LIST OF FIGURES

Figure Page

1.1 Illustration of Mixed Raster Content (MRC) document compression standard mode 1 structure. An image is divided into three layers: a binary mask layer, foreground layer, and background layer. The binary mask indicates the assignment of each pixel to the foreground layer or the background layer by a "1" (black) or "0" (white), respectively. Typically, text regions are classified as foreground while picture regions are classified as background. Each layer is then encoded independently using an appropriate encoder. 2

1.2 The COS algorithm comprises two steps: blockwise segmentation and global segmentation. The parameters of the cost function used in the global segmentation are optimized in an off-line training procedure. . . 7

1.3 Illustration of a blockwise segmentation. The pixels in each block are separated into foreground ("1") or background ("0") by comparing each pixel with a threshold t. The threshold t is then selected to minimize the total sub-class variance. 9

1.4 Illustration of class definition for each block. Four segmentation result candidates are defined; original (class 0), reversed (class 1), all background (class 2), and all foreground (class 3). The final segmentation will be one of these candidates. In this example, block size is $m = 6$. 10

1.5 Illustration of cost minimization by dynamic programming. The cost minimization is iteratively performed on individual rows of blocks, and this diagram shows the dynamic programming for the i^{th} row. Each node denotes a choice of the classes from "0", "1", "2" or "3" for a particular block, and each path from the start to the end represents a possible choice of the class combinations in the row. Each path between two nodes has a cost defined in Eq. (1.3). Therefore, the goal is to find the optimal path from the start to the end with the minimum cost. 12

1.6 Illustration of global segmentation determination by selecting center regions. Since the output segmentation for each pixel is ambiguous due to the block overlap, the final COS segmentation output is specified by the $\frac{m}{2} \times \frac{m}{2}$ center region of each $m \times m$ overlapping block. 13

1.7 Illustration of flowchart of CCC algorithm. 17

Figure Page

1.8 Illustration of how the component inversion step can correct erroneous segmentations of text. (a) Original document before segmentation, (b) result of COS binary segmentation, (c) corrected segmentation after component inversion. 18

1.9 Illustration of a Bayesian segmentation model. Line segments indicate dependency between random variables. Each component CC_i has an observed feature vector, y_i, and a class label, $x_i \in \{0, 1\}$. Neighboring pairs are indicated by thick line segments. 19

1.10 Illustration of classification probability of x_i given a single neighbor x_j as a function of the distance between the two components. The solid line shows a graph of $p(x_i \neq x_j|x_j)$ while the dashed line shows a graph of $p(x_i = x_j|x_j)$. The parameters are set to $a = 0.1$ and $b = 1.0$. 23

1.11 ICM procedure for MAP estimate 24

1.12 Illustration of a multiscale-COS/CCC algorithm. Segmentation progresses from coarse to fine scales, incorporating the segmentation result from the previous coarser scale. Both COS and CCC are performed on each scale, however only COS was adapted to the multiscale scheme. 28

1.13 Illustration of MRC encoding process. After an image is separated into foreground layer and background layer, each layer is subsampled to reduce the bitrate. After the subsampling, data-filling is performed to fill data to don't-care-regions. Finally, each layer is encoded and merged to create a MRC file. 30

1.14 Illustration of the subsampling used in this study. Simple pixel averaging of do-care pixels are used. 31

1.15 Actual image of transition color enhancement due to subsampling. The subsampling sometimes enhances the transition values along the 0 and 1 boundaries. 32

1.16 Region growing based data-filling. In order to smooth transitions between do-care and do-not-care regions of the foreground and background layers, the do-not-care pixels are replaced by the average of 8-point neighboring do-care pixels. 34

1.17 Linear data-filling fills do-not-care regions with linearly changing data using two sides of the regions. 35

1.18 Illustration of scanner characterization. 38

Figure Page

1.19 Comparison of multiscale-COS/CCC, multiscale-COS/CCC/Zheng, and DjVu in trade-off between missed detection error vs. false detection error. (a) component-wise (b) pixel-wise 44

1.20 Plot of AIC prior evaluation criteria vs. number of neighbors. The plot shows the results of four different priors with different dimension of augmented feature vectors. Small values indicate a good model fit. 50

1.21 Binary masks generated from Otsu/CCC, DjVu, LuraDocument. (a) Original test image (b) Ground truth segmentation (c) Otsu/CCC (d) DjVu (e) LuraDocument 51

1.22 Binary masks generated from multiscale-COS/CCC/Zheng, COS, COS/CCC, and multiscale-COS/CCC. (a) Original test image (b) Ground truth segmentation (c) Multiscale-COS/CCC/Zheng (d) COS (e) COS/CCC (f) Multiscale-COS/CCC 52

1.23 Text regions in the binary mask. The region is 165 × 370 pixels at 400 dpi, which corresponds to 1.04 cm × 2.34 cm. (a) Original test image (b) Ground truth segmentation (c) Otsu/CCC (d) DjVu (e) LuraDocument 53

1.24 Text regions in the binary mask. The region is 165 × 370 pixels at 400 dpi, which corresponds to 1.04 cm × 2.34 cm. (a) Original test image (b) Ground truth segmentation (c) Multiscale-COS/CCC/Zheng (d) COS (e) COS/CCC (f) Multiscale-COS/CCC 54

1.25 Picture regions in the binary mask. Picture region is 1516 × 1003 pixels at 400 dpi, which corresponds to 9.63 cm × 6.35 cm. (a) Original test image (b) Ground truth segmentation (c) Otsu/CCC (d) DjVu (e) LuraDocument 55

1.26 Picture regions in the binary mask. Picture region is 1516 × 1003 pixels at 400 dpi, which corresponds to 9.63 cm × 6.35 cm. (a) Original test image (b) Ground truth segmentation (c) Multiscale-COS/CCC/Zheng (d) COS (e) COS/CCC (f) Multiscale-COS/CCC 56

1.27 Decoded MRC image of text regions (400 dpi). (a) Original test image (b) Ground truth (300:1 compression) (c) Otsu/CCC (311:1) (e) DjVu (281:1) (f) LuraDocument (242:1) 57

1.28 Decoded MRC image of text regions (400 dpi). (a) Original test image (b) Ground truth (300:1 compression) (c) Multiscale-COS/CCC/Zheng (295:1) (d) COS (244:1) (e) COS/CCC (300:1) (f) Multiscale-COS/CCC (289:1). 58

Figure Page

1.29 Decoded MRC image of picture regions (400 dpi). (a) Original test image (b) Ground truth (300:1 compression) (c) Otsu/CCC (311:1) (d) DjVu (281:1) (e) LuraDocument (242:1) 59

1.30 Decoded MRC image of picture regions (400 dpi). (a) Original test image (b) Ground truth (300:1 compression) (c) Multiscale-COS/CCC/Zheng (295:1) (d) COS (244:1) (e) COS/CCC (300:1) (f) Multiscale-COS/CCC (289:1). 60

1.31 The yellowish or redish components were classified as text by the CCC algorithm, whereas the bluish components were classified as non-text. The brightness of each connected component indicates the intensity of the conditional probability which is described as $P(x_i|x_{\partial i})$. 61

2.1 Scanned image examples of a newspaper 64

2.2 Flowchart of region growing auto cropping 67

2.3 Region growing for auto cropping 68

2.4 Examples of region growing auto cropping 69

2.5 Flowchart of convex-hull cropping 70

2.6 Example of convex-hull cropping 71

2.7 Final result of automatic cropping 71

2.8 Paper-white estimation flowchart 72

2.9 Region growing for paper-white estimation 73

2.10 Black colorant estimation flowchart 75

2.11 (a) EPSON scanned image (b) HP scanned image (c) Original Samsung scanned image with no contrast adjustment (d) Processed Samsung scanned image with automatic contrast adjustment for the first example of newspaper materials. 80

2.12 (a) EPSON scanned image (b) HP scanned image (c) Original Samsung scanned image with no contrast adjustment (d) Processed Samsung scanned image with automatic contrast adjustment for the second example of newspaper materials. 81

2.13 (a) EPSON scanned image (b) HP scanned image (c) Original Samsung scanned image with no contrast adjustment (d) Processed Samsung scanned image with automatic contrast adjustment for the first example of magazine materials. 82

Figure Page

2.14 (a) EPSON scanned image (b) HP scanned image (c) Original Samsung scanned image with no contrast adjustment (d) Processed Samsung scanned image with automatic contrast adjustment for the second example of magazine materials. 83

2.15 The first example of show-through blocking for Samsung scanned image. 84

2.16 The second example of show-through blocking for Samsung scanned image. 84

SYMBOLS

$O_{i,j}$	indexed overlapping block of an image
M	number of overlapping blocks in vertical direction
N	number of overlapping blocks in horizontal direction
$m \times m$	number of pixels of each overlapping block
$\tilde{O}_{i,j}$	gray image formed from $O_{i,j}$ by selecting the color axis which has the largest variance
$\eta^2_{i,j}$	total sub-class variance of foreground ("1") and background ("0") within a block
$N_{0,i,j}$	number of pixels classified as 0 in $\tilde{O}_{i,j}$
$N_{1,i,j}$	number of pixels classified as 1 in $\tilde{O}_{i,j}$
$\sigma^2_{0,i,j}$	variance within pixels classified as 0 in $\tilde{O}_{i,j}$
$\sigma^2_{1,i,j}$	variance within pixels classified as 1 in $\tilde{O}_{i,j}$
$\sigma_{i,j}$	standard deviation of all the pixels in a block
$\gamma^2_{i,j}$	minimum value of $\eta^2_{i,j}$
$s_{i,j}$	segmentation class of $O_{i,j}$
$C_{i,j}$	segmented block which assigns a binary value to each pixel in $O_{i,j}$
$\tilde{C}_{i,j}$	final COS segmentation for block (i, j)
$\lambda_1 \ldots \lambda_4$	weight coefficients of a cost function
$\mathcal{E}$	square root of the total sub-class variation within a block given the assumed segmentation
V_1	number of segmentation mismatches between pixels in the overlapping region between block $\tilde{C}_{i,j}$ and the horizontally adjacent block $\tilde{C}_{i,j-1}$

V_2	number of segmentation mismatches between pixels in the overlapping region between block $\tilde{C}_{i,j}$ and the vertically adjacent block $\tilde{C}_{i-1,j}$
V_3	number of the pixels classified as foreground (i.e. "1") in $\tilde{C}_{i,j}$
V_4	number of mismatched pixels within a block between the current scale segmentation and the previous coarser scale segmentation
$B_{i,j}$	A set of pixels in a block at the position (i, j)
f_1	Cost function of COS algorithm for a single scale scheme
f_2	Cost function of COS algorithm for a multiscale scheme
$(\cdot)^{(n)}$	term for the n^{th} scale
$N_{missed_detection}$	number of pixels for missed detection
$N_{false_detection}$	number of pixels for false detection
ω	weighting coefficient between missed detection and false detection in error
N_{gt}	number of connected components in the ground truth
N_d	number of detected connected components
N_{fa}	number of false components
X_{MP}	number of pixels in missed detections
X_{FP}	number of pixels in false detections
X_{TOTAL}	total number of pixels in ground truth
p_{MC}	fraction of missed components
p_{FC}	fraction of false components
p_{MP}	fraction of missed detections of individual pixels
p_{FP}	fraction of false detections of individual pixels

ABBREVIATIONS

MRC	Mixed Raster Content
COS	Cost Optimized Segmentation
CCC	Connected Component Classification
JPEG	Joint Photographic Experts Group
JBIG	Joint Bi-level Image experts Group
OCR	Optical Character Recognition
DCT	Discrete Cosine Transform
MRF	Markov Random Field
MAP	Maximum a Posteriori
ICM	Iterative Conditional Modes
ML	Maximum Likelihood

ABSTRACT

Haneda, Eri Ph.D., Purdue University, May 2011. Markov Random Field Model Based Text Segmentation and Image Post Processing of Complex Scanned Documents. Major Professor: Charles A. Bouman.

In this dissertation, two image processing studies will be presented. The first study is segmentation for MRC document compression using an MRF model, and the second study is an automatic contrast enhancement scheme for a digital image capture device.

In the first study, we developed a new document segmentation scheme for Mixed Raster Content (MRC) standard. The Mixed Raster Content standard (ITU-T T.44) specifies a framework for document compression which can dramatically improve the compression/quality tradeoff as compared to traditional lossy image compression algorithms. The key to MRC compression is the separation of the document into foreground and background layers, represented as a binary mask. Therefore, the resulting quality and compression ratio of a MRC document encoder is highly dependent on the segmentation algorithm used to compute the binary mask.

In this study, we propose a novel multiscale segmentation scheme for MRC document encoding based on the sequential application of two algorithms. The first algorithm, cost optimized segmentation (COS), is a blockwise segmentation algorithm formulated in a global cost optimization framework. The second algorithm, connected component classification (CCC), refines the initial segmentation by classifying feature vectors of connected components using an Markov random field (MRF) model. The combined COS/CCC segmentation algorithms are then incorporated into a multiscale framework in order to improve the segmentation accuracy of text with varying size. In comparisons to state-of-the-art commercial MRC products and se-

lected segmentation algorithms in the literature, we show that the new algorithm achieves greater accuracy of text detection but with a lower false detection rate of non-text features. We also demonstrate that the proposed segmentation algorithm can improve the quality of decoded documents while simultaneously lowering the bit rate.

In the second study, we developed a robust algorithm to perform automatic contrast enhancement. The motivation for this study is that the background of scanned images by digital scanners sometimes appear too dark. This is particularly true for scans of paper materials such as newspapers, magazines and phone books. In such cases, a lighter and more uniform background color is typically preferred because it has the advantages that the contrast of the image is more distinct, the background noise is reduced, and the background color is more consistent across paper materials.

Our objective of this study is to snap the background color of the paper material to display-white and the full black colorant of the printer to display black without large color shift. In addition, we want to reduce show-through effects of thin paper materials. Our algorithm described in this document consists of four components: auto cropping, paper-white estimation, paper-black estimation, and linear contrast stretch. First, the algorithm performs auto cropping of an input image to extract the region which contains only paper. Next, paper-white color and black colorant color are estimated using the maximum and minimum values for each RGB channel. Finally, a linear contrast stretch is performed by snapping the estimated paper-white and estimated black colorant to the largest and smallest encoded value to increase the dynamic range. We show quantitative evaluation for our paper-white estimation algorithm, and show qualitative results of overall contrast stretch.

1. SEGMENTATION FOR MRC DOCUMENT COMPRESSION USING A MRF MODEL

1.1 Introduction

With the wide use of networked equipment such as computers, scanners, printers and copiers, it has become more important to efficiently compress, store, and transfer large document files. For example, a typical color document scanned at 300 dpi requires approximately 24M bytes of storage without compression. While JPEG and JPEG2000 are frequently used tools for natural image compression, they are not very effective for the compression of raster scanned compound documents which typically contain a combination of text, graphics, and natural images. This is because the use of a fixed DCT or wavelet transformation for all content typically results in severe ringing distortion near edges and line-art.

The mixed raster content (MRC) standard is a framework for layer-based document compression defined in the ITU-T T.44 [1] that enables the preservation of text detail while reducing the bitrate of encoded raster documents. The most basic MRC approach, MRC mode 1, divides an image into three layers: a binary mask layer, foreground layer, and background layer. The binary mask indicates the assignment of each pixel to the foreground layer or the background layer by a "1" or "0" value, respectively. Typically, text regions are classified as foreground while picture regions are classified as background. Each layer is then encoded independently using an appropriate encoder. For example, foreground and background layers may be encoded using traditional photographic compression such as JPEG or JPEG2000 while the binary mask layer may be encoded using symbol-matching based compression such as JBIG or JBIG2. Note that different compression ratios and subsampling rates are often used for foreground and background layers due to their different characteristics.

Typically, the foreground layer is more aggressively compressed than the background layer because the foreground layer requires lower color and spatial resolution. Figure 1.1 shows an example of layers in an MRC mode 1 document.

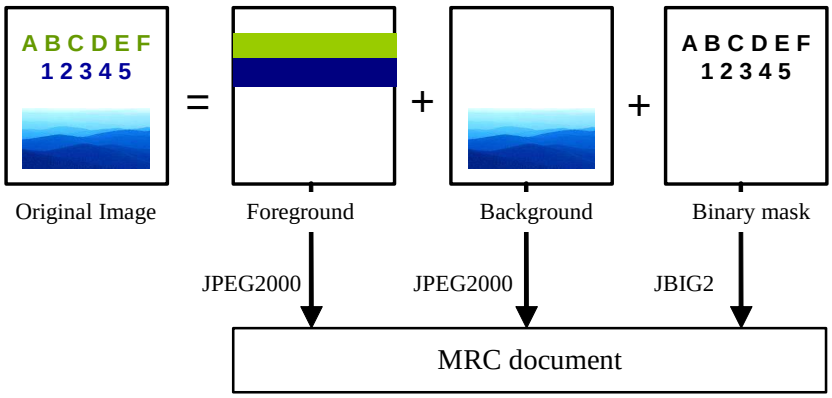


Fig. 1.1. Illustration of Mixed Raster Content (MRC) document compression standard mode 1 structure. An image is divided into three layers: a binary mask layer, foreground layer, and background layer. The binary mask indicates the assignment of each pixel to the foreground layer or the background layer by a "1" (black) or "0" (white), respectively. Typically, text regions are classified as foreground while picture regions are classified as background. Each layer is then encoded independently using an appropriate encoder.

Perhaps the most critical step in MRC encoding is the segmentation step, which creates a binary mask that separates text and line-graphics from natural image and background regions in the document. Segmentation influences both the quality and bitrate of an MRC document. For example, if a text component is not properly detected by the binary mask layer, the text edges will be blurred by the background layer encoder. Alternatively, if non-text is erroneously detected as text, this error can also cause distortion through the introduction of false edge artifacts and the excessive smoothing of regions assigned to the foreground layer. Furthermore, erroneously detected text can also increase the bit rate required for symbol-based compression methods such as JBIG2. This is because erroneously detected and unstructured non-text symbols are not be efficiently represented by JBIG2 symbol dictionaries.

Many segmentation algorithms have been proposed for accurate text extraction, typically with the application of optical character recognition (OCR) in mind. One of the most popular top-down approaches to document segmentation is the X-Y cut algorithm [2] which works by detecting white space using horizontal and vertical projections. The run length smearing algorithm (RLSA) [3,4] is a bottom-up approach which basically uses region growing of characters to detect text regions, and the Docstrum algorithm proposed in [5] is another bottom-up method which uses k-nearest neighbor clustering of connected components. Chen et. al recently developed a multi-plane based segmentation method by incorporating a thresholding method [6]. A summary of the algorithms for page segmentation can be found in [7–11].

Perhaps the most traditional approach to document binarization is Otsu's method [12] which thresholds pixels in an effort to divide the document's histogram into objects and background. There are many modified versions of Otsu's method [6, 13]. While Otsu uses a global thresholding approach, Niblack [14] and Sauvola [15] use a local thresholding approach. Kapur's method [16] uses entropy information for the global thresholding, and Tsai [17] uses a moment preserving approach. A comparison of the algorithms for text segmentation can be found in [18].

In order to improve text extraction accuracy, some text segmentation approaches also use character properties such as size, stroke width, directions, and run-length histogram [19–21]. Other segmentation approaches for document coding have used rate-distortion minimization as a criteria for document binarization [22, 23].

Many recent approaches to text segmentation have been based on statistical models. One of the best commercial text segmentation algorithms, which is incorporated in the DjVu document encoder, uses a hidden Markov model (HMM) [24, 25]. The DjVu software package is perhaps the most popular MRC-based commercial document encoder. Although there are other MRC-based encoders such as LuraDocument [26], we have found DjVu to be the most accurate and robust algorithm available for document compression. However, as a commercial package, the full details of the DjVu algorithm are not available. The use of Markov random field (MRF) models, which is

more generalized statistical framework, have also been explored for image segmentation in the past [27]. Zheng *et al.* [28] used an MRF model to exploit the contextual document information for noise removal. Similarly, Kumar [29] *et al.* used an MRF model to refine the initial segmentation generated by the wavelet analysis. J. G. Kuk *et al.* and Cao *et al.* also developed a MAP-MRF text segmentation framework which incorporates their proposed prior model [30, 31].

Recently, a conditional random field (CRF) model, originally proposed by Lafferty [32], has attracted interest as an improved model for segmentation. The CRF model differs from the traditional MRF models in that it directly models the posterior distribution of labels given observations. For this reason, in the CRF approach the interactions between labels are a function of both labels and observations. The CRF model has been applied to different types of labeling problems including blockwise segmentation of manmade structures [33], natural image segmentation [34], and pixelwise text segmentation [35].

Perhaps one of the greatest challenges in document segmentation for MRC is the correct assignment of segmented regions to foreground and background [36]. A frequently used approach is to assign lighter colors to background and darker colors to the foreground [22, 37]. However, this assumption is not valid for light colored text on a dark background. Another challenge is detecting text of different size simultaneously. We frequently encounter size limitation issues for text segmentation. Segmentation is further complicated when the background is graded or noisy. These features are often contained in flyers, magazines, and newspapers.

In this document, we present a robust multiscale segmentation algorithm for both detecting and segmenting text in a complex document containing background gradation, varying text size, reversed contrast text, and noisy backgrounds. While considerable research has been done in the area of text segmentation, our approach differs in that it integrates a stochastic model of text structure and context into a multiscale framework in order to best meet the requirements of MRC document compression. In particular, our method is designed to minimize false detections of unstructured non-

text components (which can create artifacts and increase bit-rate) while accurately segmenting true-text components of varying size and with varying backgrounds. Using this approach, our algorithm can achieve higher decoded image quality at a lower bit-rate than generic algorithms for document segmentation. We note that a preliminary version of this approach, without the use of an MRF prior model, was presented in the conference paper of [38], and that the source code for the method described in this document is publicly available. [1]

Our segmentation method is composed of two algorithms that are applied in sequence: the cost optimized segmentation (COS) algorithm and the connected component classification (CCC) algorithm. The COS algorithm is a blockwise segmentation algorithm based on cost optimization. The COS produces a binary image from a gray level or color document; however, the resulting binary image typically contains many false text detections. The CCC algorithm further processes the resulting binary image to improve the accuracy of the segmentation. It does this by detecting non-text components (i.e. false text detections) in a Bayesian framework which incorporates an Markov random field (MRF) model of the component labels. One important innovation of our method is in the design of the MRF prior model used in the CCC detection of text components. In particular, we design the energy terms in the MRF distribution so that they adapt to attributes of the neighboring components' relative locations and appearance. By doing this, the MRF can enforce stronger dependencies between components which are more likely to have come from related portions of the document. Both the COS and CCC algorithms are also formulated in a multiscale framework to improve detection accuracy for both small and large text. Our experimental results indicate that the multiscale COS/CCC algorithm achieves greater accuracy of text detection but with a lower false detection rate of non-text, as compared to state-of-the-art commercial MRC products.

The organization of this thesis is as follows. In section 1.2 and section 1.3, COS and CCC algorithms are described. In section 1.4, the multiscale implementation for COS

[1] https://engineering.purdue.edu/~bouman

and CCC is described. The section 1.5 explains the MRC encoding procedure used in this study. The section 1.6 presents experimental results, in both quantitative and qualitative ways. The results include a comparison with other commercial products or other popular segmentation algorithms in terms of segmentation accuracy, the resulting bitrate of the binary mask, and computational speed. The MRC decoded image comparison is also shown in this section. Finally, the section 1.7 describes the summary of this research.

1.2 Cost Optimized Segmentation (COS)

The Cost Optimized Segmentation (COS) algorithm is a block based segmentation algorithm formulated as a global cost optimization problem. The COS algorithm is comprised of two components: blockwise segmentation and global segmentation. The blockwise segmentation divides the input image into overlapping blocks and produces an initial segmentation for each block. The global segmentation is then computed from the initial segmented blocks so as to minimize a global cost function, which is carefully designed to favor segmentations that capture text components. The parameters of the cost function are optimized in an off-line training procedure. A block diagram for COS is shown in Figure 1.2.

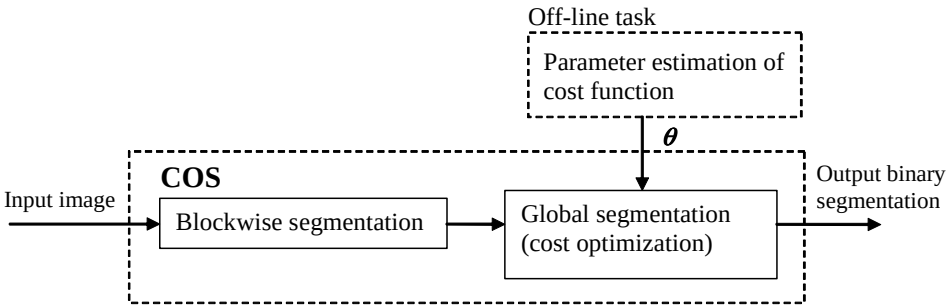


Fig. 1.2. The COS algorithm comprises two steps: blockwise segmentation and global segmentation. The parameters of the cost function used in the global segmentation are optimized in an off-line training procedure.

1.2.1 Blockwise segmentation

Blockwise segmentation is performed by first dividing the image into overlapping blocks, where each block contains $m \times m$ pixels, and adjacent blocks overlap by $m/2$ pixels in both the horizontal and vertical directions. The blocks are denoted by $O_{i,j}$ for $i = 1, .., M$, and $j = 1, .., N$, where M and N are the number of the blocks in

the vertical and horizontal directions, respectively. If the height and width of the input image is not divisible by m, the image is padded with zeros. For each block, the color axis having the largest variance over the block is selected and stored in a corresponding gray image block, $\tilde{O}_{i,j}$.

The pixels in each block are segmented into foreground ("1") or background ("0") by the clustering method of Cheng and Bouman [23]. The clustering method classifies each pixel in $\tilde{O}_{i,j}$ by comparing it to a threshold t. This threshold is selected to minimize the total sub-class variance. More specifically, the minimum value of the total sub-class variance is given by

$$\gamma_{i,j}^2 = \min_{t\in[0,255]} \frac{N_{0,i,j} * \sigma_{0,i,j}^2 + N_{1,i,j} * \sigma_{1,i,j}^2}{N_{0,i,j} + N_{1,i,j}} \tag{1.1}$$

where $N_{0,i,j}$ and $N_{1,i,j}$ are number of pixels classified as 0 and 1 in $\tilde{O}_{i,j}$ by the threshold t, and $\sigma_{0,i,j}^2$ and $\sigma_{1,i,j}^2$ are the variances within each sub-group (See Figure 1.3). Note that the sub-class variance can be calculated efficiently. First, we create a histogram by counting the number of pixels which fall into each value between 0 and 255. For each threshold $t \in [0, 255]$, we can recursively calculate $\sigma_{0,i,j}^2$ and $\sigma_{1,i,j}^2$ from the values calculated for the previous threshold of $t - 1$. The threshold that minimizes the sub-class variance is then used to produce a binary segmentation of the block denoted by $C_{i,j} \in \{0, 1\}^{m\times m}$.

1.2.2 Global segmentation

The global segmentation step integrates the individual segmentations of each block into a single consistent segmentation of the page. To do this, we allow each block to be modified using a class assignment denoted by, $s_{i,j} \in \{0, 1, 2, 3\}$.

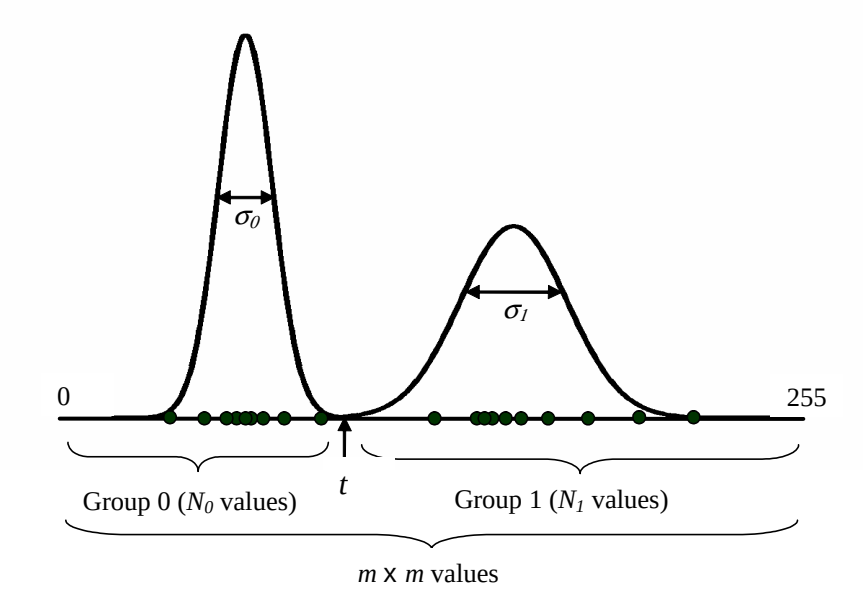


Fig. 1.3. Illustration of a blockwise segmentation. The pixels in each block are separated into foreground ("1") or background ("0") by comparing each pixel with a threshold t. The threshold t is then selected to minimize the total sub-class variance.

$$\begin{aligned}
s_{i,j} = 0 \quad &\Rightarrow \quad \tilde{C}_{i,j} = C_{i,j} \qquad \text{(Original)} \\
s_{i,j} = 1 \quad &\Rightarrow \quad \tilde{C}_{i,j} = \neg C_{i,j} \qquad \text{(Reversed)} \\
s_{i,j} = 2 \quad &\Rightarrow \quad \tilde{C}_{i,j} = \{0\}^{m \times m} \quad \text{(All background)} \\
s_{i,j} = 3 \quad &\Rightarrow \quad \tilde{C}_{i,j} = \{1\}^{m \times m} \quad \text{(All foreground)}
\end{aligned} \tag{1.2}$$

Notice that for each block, the four possible values of $s_{i,j}$ correspond to four possible changes in the block's segmentation: original, reversed, all background, or all foreground. If the block class is "original", then the original binary segmentation of the block is retained. If the block class is "reversed", then the assignment of each pixel in the block is reversed (i.e. 1 goes to 0, or 0 goes to 1). If the block class is set to "all background" or "all foreground", then the pixels in the block are set to all 0's or all 1's, respectively. Figure 1.4 illustrates an example of the four possible classes where black indicates a label of "1" (foreground) and white indicates a label of "0" (background).

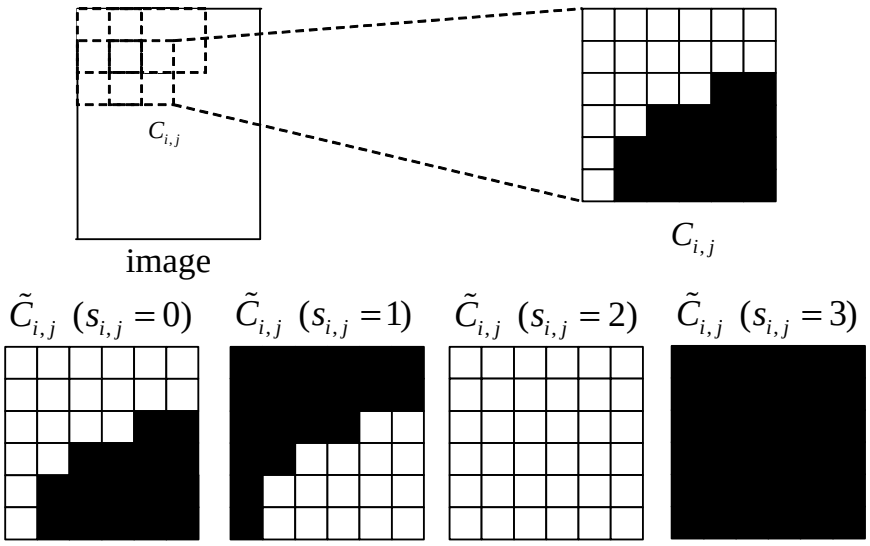


Fig. 1.4. Illustration of class definition for each block. Four segmentation result candidates are defined; original (class 0), reversed (class 1), all background (class 2), and all foreground (class 3). The final segmentation will be one of these candidates. In this example, block size is $m = 6$.

Our objective is then to select the class assignments, $s_{i,j} \in \{0, 1, 2, 3\}$, so that the resulting binary masks, $\tilde{C}_{i,j}$, are consistent. We do this by minimizing the following global cost as a function of the class assignments, $S = [s_{i,j}]$ for all i, j,

$$f_1(S) = \sum_{i=1}^{M} \sum_{j=1}^{N} \{\mathcal{E}(s_{i,j}) + \lambda_1 V_1(s_{i,j}, s_{i,j+1}) + \lambda_2 V_2(s_{i,j}, s_{i+1,j}) + \lambda_3 V_3(s_{i,j})\} . \quad (1.3)$$

As it is shown, the cost function contains four terms, the first term representing the fit of the segmentation to the image pixels, and the next three terms representing regularizing constraints on the segmentation. The values λ_1, λ_2, and λ_3 are then model parameters which can be adjusted to achieve the best segmentation quality.

The first term $\mathcal{E}$ is the square root of the total sub-class variation within a block given the assumed segmentation. More specifically,

$$\mathcal{E}(s_{i,j}) = \begin{cases} \gamma_{i,j} & \text{if } s_{i,j} = 0 \text{ or } s_{i,j} = 1 \\ \sigma_{i,j} & \text{if } s_{i,j} = 2 \text{ or } s_{i,j} = 3 \end{cases} \quad (1.4)$$

where $\sigma_{i,j}$ is the standard deviation of all the pixels in the block. Since $\gamma_{i,j}$ must always be less than or equal to $\sigma_{i,j}$, the term $\mathcal{E}$ can always be reduced by choosing a finer segmentation corresponding to $s_{i,j} = 0$ or 1 rather than smoother segmentation corresponding to $s_{i,j} = 2$ or 3.

The terms V_1 and V_2 regularize the segmentation by penalizing excessive spatial variation in the segmentation. To compute the term V_1, the number of segmentation mismatches between pixels in the overlapping region between block $\tilde{C}_{i,j}$ and the horizontally adjacent block $\tilde{C}_{i,j+1}$ is counted. The term V_1 is then calculated as the number of the segmentation mismatches divided by the total number of pixels in the overlapping region. Also V_2 is similarly defined for vertical mismatches. By minimizing these terms, the segmentation of each block is made consistent with neighboring blocks.

The term V_3 denotes the number of the pixels classified as foreground (i.e. "1") in $\tilde{C}_{i,j}$ divided by the total number of pixels in the block. This cost penalizes segmentations that assign too many pixels to the foreground to ensure that most of the area of image is classified as background. Therefore, this cost ensures that most of the area of image is classified as background.

For computational tractability, the cost minimization is iteratively performed on individual rows of blocks, using a dynamic programming approach [39]. Figure 1.5 shows a diagram of the dynamic programming process for the i^{th} row of image blocks. Each node denotes a choice of the classes from "0", "1", "2" or "3" for a particular block. Each path from the start to the end represents a possible choice of the class combinations in the row. Each path between two nodes has a cost defined in Eq. (1.3). Therefore, our goal is equivalent to find the optimal path from the start to the end with the minimum cost.

Note that row-wise approach does not generally minimize the global cost function in one pass through the image. Therefore, multiple iterations are performed from top to bottom in order to adequately incorporate the vertical consistency term. In the first iteration, the optimization of i^{th} row incorporates the V_2 term containing only

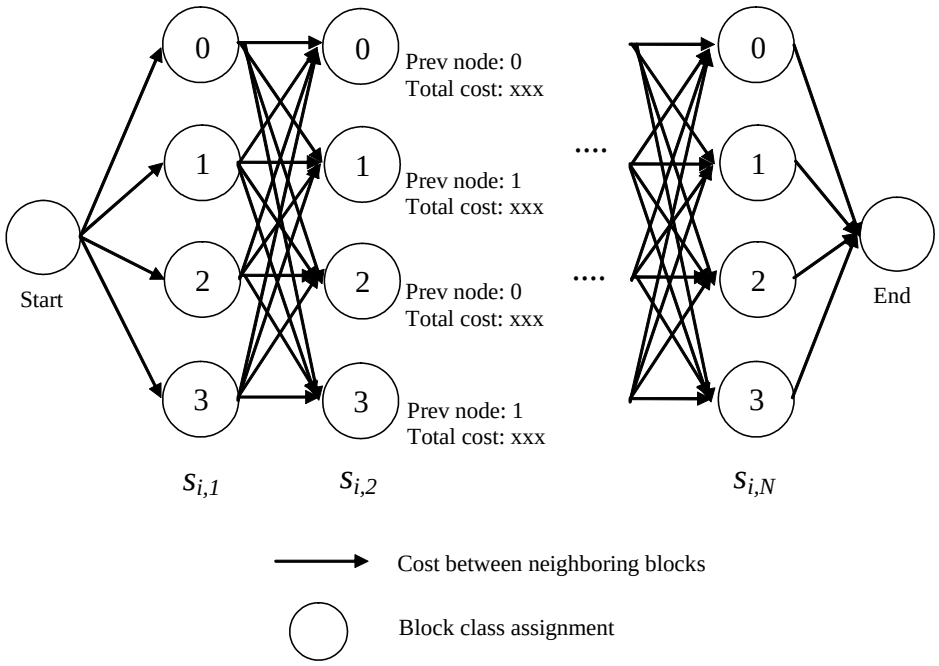


Fig. 1.5. Illustration of cost minimization by dynamic programming. The cost minimization is iteratively performed on individual rows of blocks, and this diagram shows the dynamic programming for the i^{th} row. Each node denotes a choice of the classes from "0", "1", "2" or "3" for a particular block, and each path from the start to the end represents a possible choice of the class combinations in the row. Each path between two nodes has a cost defined in Eq. (1.3). Therefore, the goal is to find the optimal path from the start to the end with the minimum cost.

the $i-1^{th}$ row. Starting from the second iteration, V_2 terms for both the $i-1^{th}$ row and $i+1^{th}$ row are included. The optimization stops when no changes occur to any of the block classes. This optimization is sometimes trapped in local minima, but it does approximately minimize the cost function. Experimentally, the sequence of updates typically converges within 20 iterations.

The cost optimization produces a set of classes for overlapping blocks. Since the output segmentation for each pixel is ambiguous due to the block overlap, the final

COS segmentation output is specified by the $\frac{m}{2} \times \frac{m}{2}$ center region of each $m \times m$ overlapping block. (See Figure 1.6)

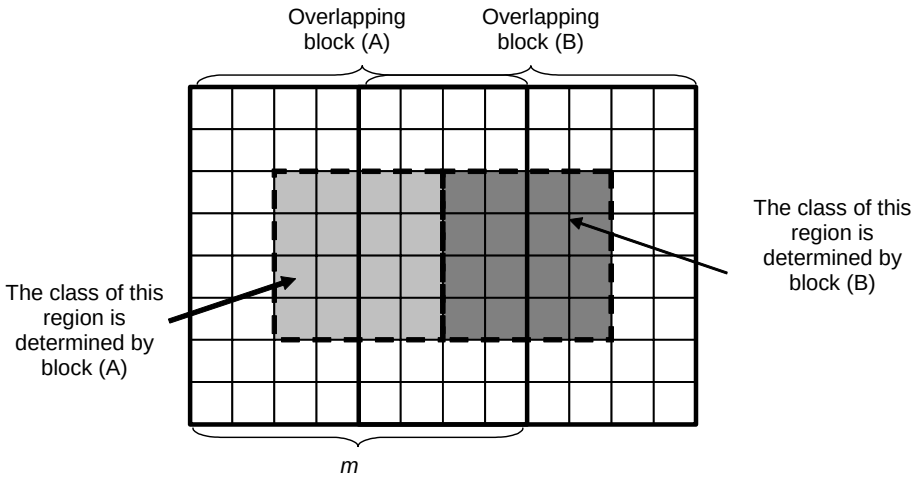


Fig. 1.6. Illustration of global segmentation determination by selecting center regions. Since the output segmentation for each pixel is ambiguous due to the block overlap, the final COS segmentation output is specified by the $\frac{m}{2} \times \frac{m}{2}$ center region of each $m \times m$ overlapping block.

The weighting coefficients λ_1,λ_2, and λ_3 were found by minimizing the weighted pixel error between segmentation results of training images and corresponding ground truth segmentations. A ground truth segmentation was generated manually using an image editor by creating a mask that indicates the text in the image. The weighted error criteria which we minimized is given by

$$\varepsilon_{weighted} = (1-\omega)N_{missed_detection} + \omega N_{false_detection} \tag{1.5}$$

where $\omega \in [0,1]$, and the terms $N_{missed_detection}$ and $N_{false_detection}$ are the number of pixels in the missed detection and false detection categories, respectively. For our application, the missed detections are generally more serious than false detections, so we used a value of $\omega = 0.09$ which more heavily weighted miss detections.

Although the COS segmentation is a robust document segmentation algorithm, we found that the COS algorithm is still incomplete in two aspects of segmentation accuracy even if the optimized parameters are used. First, the COS algorithm is not sensitive enough for text embedded in noisy background. Second, it often classifies the sharp edges in picture regions as text components. The COS segmentation does not generally show the best performance in the trade-off between missing detections and false detections. This is probably because the COS algorithm operates blockwise, and therefore cannot always exploit the text properties such as stroke, shape, appearance, and similarity to neighboring text components. Therefore, our approach is to adjust the weighting coefficients of the cost function in COS to be overly sensitive for text detection, then remove the redundant detection of sharp edges using the text connected component information described above. In the next section, we will show how the resulting false detections can be effectively reduced.

1.3 Connected Component Classification (CCC)

1.3.1 Markov random field model

A Markov random field (MRF) is a model for a joint distribution, which is often used when interactions occur between neighboring elements. The MRF model simplifies the joint distribution by using contextual dependency of neighboring elements. More specifically, an MRF satisfies the Markov property:

$$p(x_s|x_r \ for \ r \neq s) = p(x_s|x_{\partial s}) \tag{1.6}$$

where x_s represents the label of the element s, and $x_{\partial s}$ represents the labels for the set of neighbors of element s. This model drastically simplifies the joint distribution and MAP optimization due to the MRF-Gibbs equivalence theorem, which was established by Hammersley and Clifford [40].

According to the Hammersley-Clifford theorem, the joint distribution of the labels may be expressed as a Gibbs distribution if an MRF model is assumed [41]. A Gibbs distribution takes the following form

$$p(x) = \frac{1}{Z} \exp\left\{-\frac{1}{T}U(x)\right\} \tag{1.7}$$

where $p(x)$ is a joint distribution, the Z is a normalizing constant called the partition function, and the T is a constant called the temperature which may be assumed to be 1. The term $U(x)$ is called the energy function, which is expressed as a sum of clique potentials $V_c(x)$ over the set of all cliques, C.

$$U(x) = \sum_{c \in C} V_c(x) \tag{1.8}$$

The clique is a set of elements which neighbor each other. In an MRF model, the neighborhood relationship must have the following properties:

- An element s is not neighboring to itself: $s \notin \partial s$
- The neighbor relationship is mutual: $s \in \partial r \Longleftrightarrow r \in \partial s$

The maximum size of cliques can be chosen arbitrarily; however the pairwise clique is the most popular choice. The pairwise cliques are the lowest order constraints, but they are widely used because of the simple form and low computation. For pixelwise labeling problems, 4-point neighborhood or 8-point neighborhood system on a rectangular lattice is often used for simplicity.

In MRF modeling, the design of the neighborhood system and energy function is a primary task since these two elements fully define the model. There are two keys in the MRF modeling for text segmentation. First, we define the neighborhood system component-wise, as opposed to pixelwise. Most traditional segmentation algorithms using MRF models employs pixelwise segmentation. However, we adopt an MRF model to component-wise segmentation, which classifies each connected component on the initial segmentation results into text or non-text. Second, our model also uses

observation data in the energy function. Whereas most traditional energy functions contain only contextual data (i.e. x_s and $x_{\partial s}$), we decided to use observation data to define the relationship between neighboring components. Recently, a conditional random field (CRF) model, originally proposed by Lafferty [32], has attracted interest as an alternative model of MRF. The CRF model differs from the traditional MRF models in that it directly models the posterior distribution of labels given observations. For this reason, in the CRF approach the interactions between labels are a function of both labels and observations. The CRF model has been applied to different types of labeling problems including blockwise segmentation of manmade structures [33], natural image segmentation [34], and pixelwise text segmentation [35].

Text extraction can be a good application for MRF models because text usually appears in clusters. Relative positions or similarities among neighboring text components such as size, shape and edge depth may be described by the MRF model for text detection. An Markov random field (MRF) is generally used in a statistical framework which models the joint probability of the observed data and the corresponding labels. Let's $y = [y_1, y_2, \ldots y_N]$ denote observation data and $x = [x_1, x_2, \ldots x_N]$ denote the corresponding labels. Then, the maximum a posteriori (MAP) estimate can be computed as

$$\hat{x}_{MAP} = \underset{x}{\operatorname{argmax}}\{\log p_{y|x}(y|x) + \log_x p(x)\}. \tag{1.9}$$

The first term is called data term while the second term is called the prior term where the MRF model is applied. More detailed technical discussion about connected component classification (CCC) algorithm will be given in the following sections.

1.3.2 Connected component classification (CCC) algorithm

The connected component classification (CCC) algorithm refines the segmentation produced by COS by removing many of the erroneously detected non-text components. While the COS algorithm uses blockwise segmentation, CCC algorithm uses component-wise segmentation. The CCC algorithm proceeds in three steps:

connected component extraction, component inversion, and finally component classification. A flowchart for connected component classification is shown in Figure 1.7.

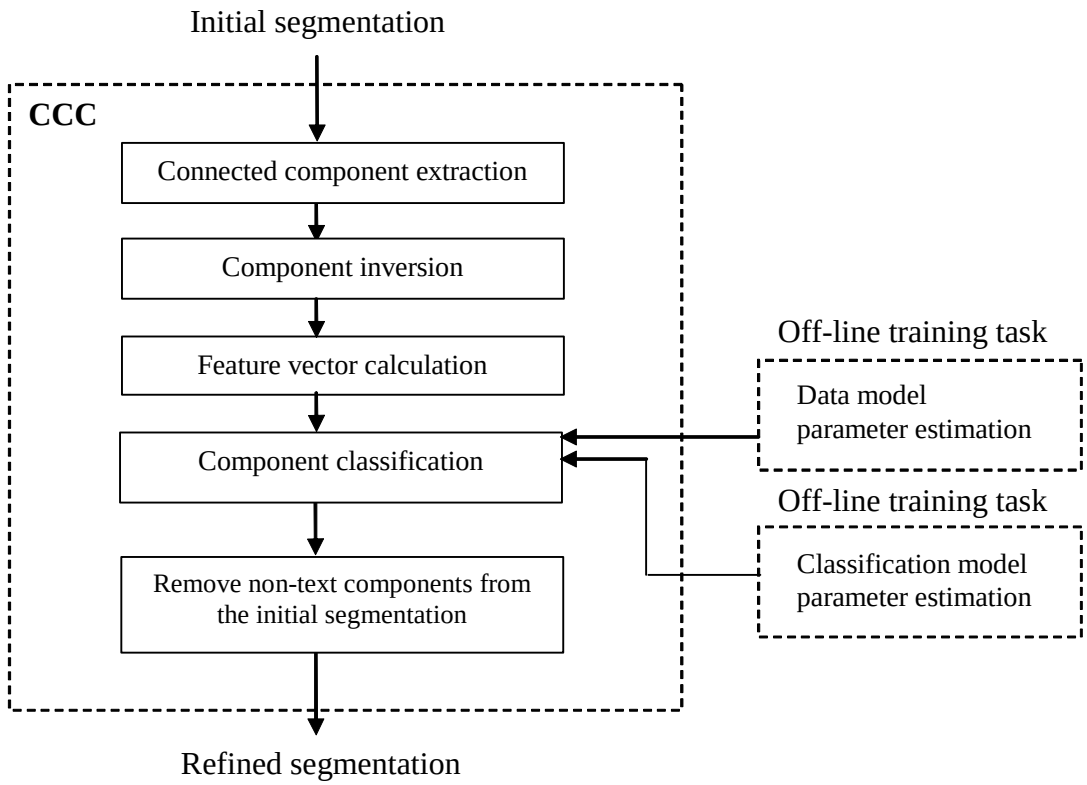
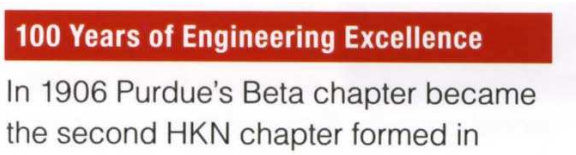


Fig. 1.7. Illustration of flowchart of CCC algorithm.

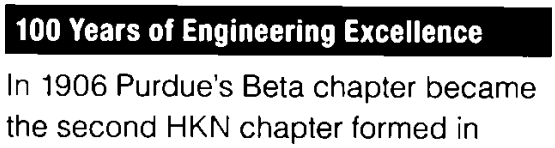
The connected component extraction step first identifies all connected components in the COS binary segmentation using a 4-point neighborhood. In this case, connected components less than six pixels were ignored because they are nearly invisible at 300 dpi resolution. The component inversion step corrects text segmentation errors that sometimes occur in COS segmentation when text is locally embedded in a highlighted region (See Figure 1.8 (a)). Figure 1.8 (b) illustrates this type of error where text is initially segmented as background. Notice the text "100 Years of Engineering Excellence" is initially segmented as background due to the red surrounding region.

In order to correct these errors, we first detect foreground components that contain more than eight interior background components (holes). In each case, if the total number of interior background pixels is less than half of the surrounding foreground pixels, the foreground and background assignments are inverted. Figure 1.8 (c) shows the result of this inversion process. Note that this type of error is a rare occurrence in the COS segmentation.

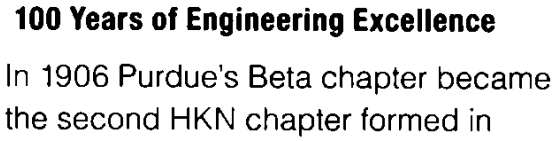
The final step of component classification is performed by extracting a feature vector for each component, and then computing a MAP estimate of the component label. The feature vector, y_i, is calculated for each connected component, CC_i, in the COS segmentation. Each y_i is a 4 dimensional feature vector which describes aspects of the i^{th} connected component including edge depth and color uniformity. Finally, the feature vector y_i is used to determine the class label, x_i, which takes a value of 0 for non-text and 1 for text.



(a) Original image



(b) Initial segmentation



(c) Preprocessed segmentation

Fig. 1.8. Illustration of how the component inversion step can correct erroneous segmentations of text. (a) Original document before segmentation, (b) result of COS binary segmentation, (c) corrected segmentation after component inversion.

The Bayesian segmentation model used for the CCC algorithm is shown in Figure 1.9. The conditional distribution of the feature vector y_i given x_i is modeled by a multivariate Gaussian mixture while the underlying true segmentation labels are modeled by a Markov random field (MRF). Using this model, we classify each component by calculating the MAP estimate of the labels, x_i, given the feature vectors, y_i. In order to do this, we first determine which components are neighbors in the MRF. This is done based on the geometric distance between components on the page.

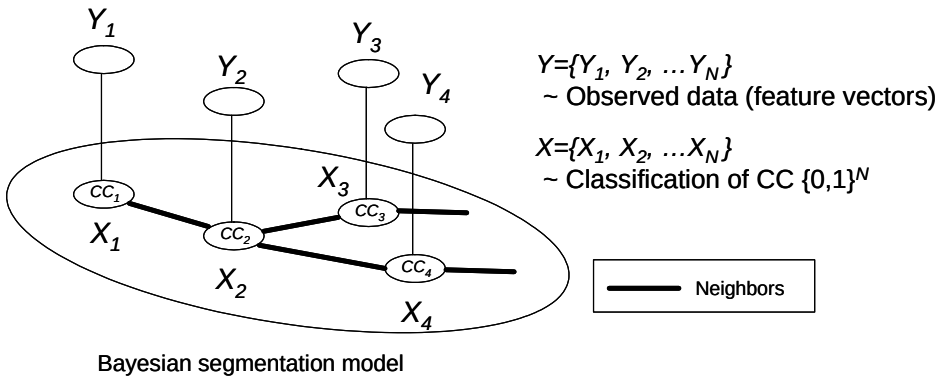


Fig. 1.9. Illustration of a Bayesian segmentation model. Line segments indicate dependency between random variables. Each component CC_i has an observed feature vector, y_i, and a class label, $x_i \in \{0, 1\}$. Neighboring pairs are indicated by thick line segments.

1.3.3 Statistical model

Here, we describe more details of the statistical model used for the CCC algorithm. The feature vectors for "text" and "non-text" groups are modeled as D-dimensional multivariate Gaussian mixture distributions,

$$p(y_i|x_i) = \sum_{m=0}^{M_{x_i}-1} \frac{a_{x_i,m}}{(2\pi)^{D/2}|R_{x_i,m}|^{-1/2}} \exp\left\{-\frac{1}{2}(y_i - \mu_{x_i,m})^t R_{x_i,m}^{-1}(y_i - \mu_{x_i,m})\right\}, \quad (1.10)$$

where $x_i \in \{0, 1\}$ is a class label of non-text or text for the i^{th} connected component. The M_0 and M_1 are the number of clusters in each Gaussian mixture distribution, and the $\mu_{x_i,m}$, $R_{x_i,m}$, and $a_{x_i,m}$ are the mean, covariance matrix, and weighting coefficient of the m^{th} cluster in each distribution. In order to simplify the data model, we also assume that the values Y_i are conditionally independent given the associated values X_i.

$$p(y|x) = \prod_{i=1}^{N} p(y_i|x_i) \tag{1.11}$$

The components of the feature vectors y_i include the information describing edge depth and color uniformity of the i^{th} connected component. The edge depth is defined as the Euclidean distance between RGB values of neighboring pixels across the component boundary (defined in the initial COS segmentation). The color uniformity is associated with the variation of the pixels outside the boundary. In this experiment, we defined a feature vector with four components, $y_i = [y_{1i}\ y_{2i}\ y_{3i}\ y_{4i}]^T$, where the first two are mean and variance of the edge depth and the last two are the variance and range of external pixel values. More details are provided in the Appendix.

To use a Markov random field model (MRF), we must define a neighbor system. To do this, we first find the pixel location at the center of mass for each connected component. Then, for each connected component, we search outward in a spiral pattern until the k nearest neighbors are found. The number k is determined in an off-line training process along with other model parameters. One concern about the k-NN method is that it does not guarantee the symmetry property of the neighboring relationship, i.e. that $s \in \partial r \iff r \in \partial s$. To ensure all neighbors are mutual (which is required for an MRF), if component s is a neighbor of component r (i.e. $s \in \partial r$), we add component r to the neighbor list of component s (i.e. $r \in \partial s$) if this is not already the case.

In order to specify the distribution of the MRF, we first define augmented feature vectors. The augmented feature vector, z_i, for the i^{th} connected component consists of the feature vector y_i concatenated with the horizontal and vertical pixel location of

the connected component's center. We found the location of connected components to be extremely valuable contextual information for text detection. The size or color information may be included, however, we found that these additional information did not dramatically improve the results. For more details of the augmented feature vector, see Appendix.

Next, we define a measure of dissimilarity between connected components in terms of the Mahalanobis distance of the augmented feature vectors given by

$$d_{i,j} = \sqrt{(z_i - z_j)^T \Sigma^{-1} (z_i - z_j)} \tag{1.12}$$

where Σ is the covariance matrix of the augmented feature vectors on training data. The covariance matrix Σ is estimated using the text component data extracted from the training set. Next, the Mahalanobis distance, $d_{i,j}$, is normalized using the equations,

$$\begin{aligned} D_{i,j} &= \frac{d_{i,j}}{\frac{1}{2}(\bar{d}_{i,\partial i} + \bar{d}_{j,\partial j})} \\ \bar{d}_{i,\partial i} &= \frac{1}{|\partial i|} \sum_{k \in \partial i} d_{i,k} \\ \bar{d}_{j,\partial j} &= \frac{1}{|\partial j|} \sum_{k \in \partial j} d_{j,k} \end{aligned} \tag{1.13}$$

where $\bar{d}_{i,\partial i}$ is the averaged distance between the i^{th} connected component and all of its neighbors, and $\bar{d}_{j,\partial j}$ is similarly defined, and the term ∂i denotes neighbors of the i^{th} connected component. This normalized distance satisfies the symmetric property, that is $D_{i,j} = D_{j,i}$.

Using the defined neighborhood system, we adopted a MRF model with pair-wise cliques. Let P be the set of all (i, j) where i and j denote neighboring connected components. Then, the X_i are assumed to be distributed as

$$p(x) = \frac{1}{Z} \exp\left\{ - \sum_{\{i,j\} \in P} w_{i,j} \delta(x_i \neq x_j) \right\} \tag{1.14}$$

$$w_{i,j} = \frac{b}{D_{i,j}^{p} + a} \tag{1.15}$$

where $\delta(\cdot)$ is an indicator function taking the value 0 or 1, and a, b, and p are scalar parameters of the MRF model. As we can see, the classification probability is penalized by the number of neighboring pairs which have different classes. This number is also weighted by the term $w_{i,j}$. If there exists a similar neighbor close to a given component, the term $w_{i,j}$ becomes large since $D_{i,j}$ is small. This favors increasing the probability that the two similar neighbors have the same class.

1.3.4 Inference

Given the analytical form of prior described in the previous sections, we are now interested in finding the estimate of X. In this study, we use a maximum a posteriori (MAP) estimate. To solve the MAP estimate of X, it is convenient to derive the equivalent local representation form that allows us to understand prior model more intuitively.

$$\begin{aligned}
p(x_i|x_{\partial i}) &= p(x_i|x_{s\neq i}) \\
&= \frac{p(x_i, x_{s\neq i})}{p(x_{s\neq i})} \\
&= \frac{p(x)}{\sum_{x_i} p(x_i, x_{s\neq i})} \\
&= \frac{\exp\{-\sum_{\{t,s\}\in P} w_{t,s}\delta(x_t \neq x_s)\}}{\sum_{x_i} \exp\{-\sum_{\{t,s\}\in P} w_{t,s}\delta(x_t \neq x_s)\}} \\
&= \frac{1}{C_i} \exp\{-\sum_{j\in\partial i} w_{i,j}\delta(x_i \neq x_j)\}
\end{aligned} \tag{1.16}$$

where the ∂i is neighbors of i^{th} component. The normalization factor C_i is defined as:

$$C_i = \sum_{x_i \in \{0,1\}} \exp\{-\sum_{j \in \partial i} w_{i,j} \delta(x_i \neq x_j)\}. \tag{1.17}$$

Figure 1.10 illustrates the classification probability of x_i given a single neighbor x_j as a function of the distance between the two components $D_{i,j}$. Here, we assume the classification of x_j is given. The solid line shows a graph of the probability $p(x_i \neq x_j|x_j)$ while the dashed line shows a graph of the probability $p(x_i = x_j|x_j)$. Note that the parameter p controls the roll-off of the function, and a and b contol the minimum and transition point for the function. The actual parameters, $\phi = [p, a, b]^T$, are optimized in an off-line training procedure (See sec. 1.3.5).

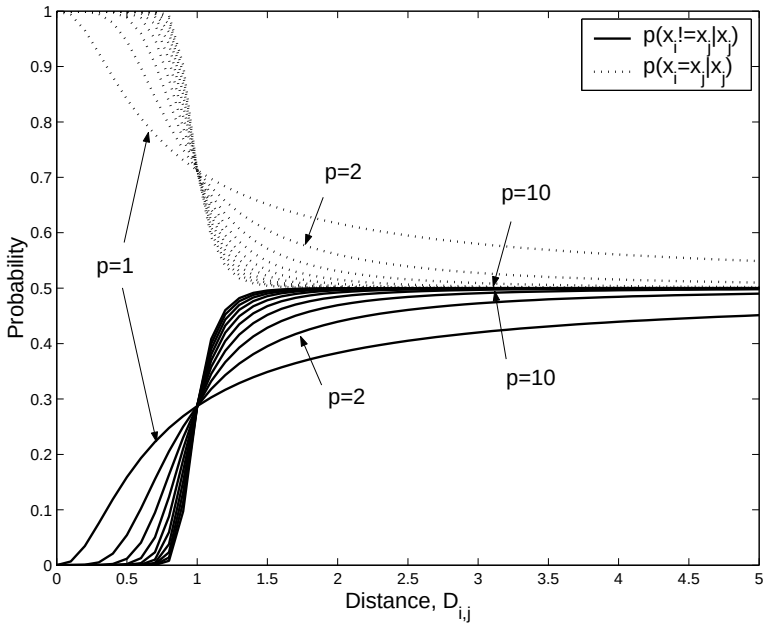


Fig. 1.10. Illustration of classification probability of x_i given a single neighbor x_j as a function of the distance between the two components. The solid line shows a graph of $p(x_i \neq x_j|x_j)$ while the dashed line shows a graph of $p(x_i = x_j|x_j)$. The parameters are set to $a = 0.1$ and $b = 1.0$.

1. Initialize $\{x_i | i \in S\}$ with ML estimation.

   For each $i \in S$,

   $x_i \leftarrow \underset{k \in \{0,1\}}{\operatorname{argmin}} \{-\log p(y_i|k)\}$

2. For each $i \in S$,

   $x_i \leftarrow \underset{k \in \{0,1\}}{\operatorname{argmin}} \{-\log p(y_i|k) + \sum_{j \in \partial i} w_{i,j}\delta(x_i \neq x_j) - c_{text}\delta(x_i = 1)\}$

3. If no change occurs to $x = [x_1\, x_2\, \ldots\, x_N]^T$, then stop.

   Otherwise go to 2.

Fig. 1.11. ICM procedure for MAP estimate

With the MRF model defined above, we can compute a maximum a posteriori (MAP) estimate to find the optimal set of classification labels $x = [x_1\, x_2\, \ldots\, x_N]^T$. The MAP estimate is given by,

$$\hat{x}_{MAP} = \underset{x \in \{0,1\}^N}{\operatorname{argmin}} \left\{ -\sum_{i \in S} \log p(y_i|x_i) + \sum_{\{i,j\} \in P} w_{i,j}\delta(x_i \neq x_j) - c_{text}\delta(x_i = 1) \right\}. \quad (1.18)$$

We introduced a term c_{text} to control the trade-off between missed and false detections. The term c_{text} may take a positive or negative value. If c_{text} is positive, both text detections and false detections increase. If it is negative, false and missed detections are reduced. If it is zero, it is a regular MAP estimate with no weights.

To find an approximate solution of (1.18), we use iterative conditional modes (ICM) which sequentially minimizes the local posterior probabilities [27, 42]. The classification labels, $\{x_i |\ i \in S\}$, are initialized with their maximum likelihood (ML) estimates, and then the ICM procedure iterates through the set of classification labels until a stable solution is reached. More specifically, the ICM procedure is given in Fig. 1.11.

1.3.5 Parameter estimation

In our statistical model, there are two sets of parameters to be estimated by a training process. The first set of parameters comes from the D-dimensional multivariate Gaussian mixture distributions given by Eq. (1.10) which models the feature vectors from text and non-text classes. The second set of parameters, $\phi = [p,\ a,\ b]^T$, controls the MRF model of Eq. (1.15). All parameters are estimated in an off-line procedure using a training image set and their ground truth segmentations. The training images were first segmented using the COS algorithm. All foreground connected components were extracted from the resulting segmentations, and then each connected component was labeled as text or non-text by matching to the components on the ground truth segmentations. The corresponding feature vectors were also calculated from the original training images.

The parameters of the Gaussian mixture distributions were estimated using the EM algorithm [43]. The EM algorithm can find a maximum likelihood (ML) estimate of parameters of a given model when we have hidden variables. In the Guassian mixture distribution, the assignments of an input data to a sub-cluster in the distribution is unknown. The EM algorithm does not find an actual assignments, but it calculates the conditional probability of each assignments to find the maximum likelihood estimate. The number of sub-clusters in each Gaussian mixture for text and non-text were also determined using the minimum description length (MDL) estimator [44]. The MDL estimator is similar to ML estimator but the penalty for overfitting to the model is added to the cost function.

The prior model parameters $\phi = [p,\ a,\ b]^T$ were independently estimated using pseudolikelihood maximization [40, 45, 46]. One of the approaches for the parameter estimation is to use a maximum likelihood (ML) estimate. However, the partition function has an intractable form and it cannot be computed easily. The pseudolikelihood estimation gives tractable form for the estimation by assuming local dependencies for conditional probability. In order to apply pseudolikelihood estimation,

we must first calculate the conditional probability of a classification label given its neighboring components' labels (See Eq. (1.16) and Eq. (1.17)). Therefore, the pseudolikelihood parameter estimation for our case is,

$$\begin{aligned}\hat{\phi} &= \underset{\phi}{\operatorname{argmax}} \prod_{i \in S} p(x_i | x_{\partial i}) \\ &= \underset{\phi}{\operatorname{argmax}} \prod_{i \in S} \frac{1}{C_i} \exp\left\{ - \sum_{j \in \partial i} w_{i,j} \delta(x_i \neq x_j) \right\} \\ &= \underset{\phi}{\operatorname{argmin}} \sum_{i \in S} \left\{ \log C_i + \sum_{j \in \partial i} w_{i,j} \delta(x_i \neq x_j) \right\}.\end{aligned} \tag{1.19}$$

In this case, C_i is easily computed. First define

$$\begin{aligned}\partial_i^0 &= \{ j \in \partial i : x_j = 0 \} \\ \partial_i^1 &= \{ j \in \partial i : x_j = 1 \},\end{aligned} \tag{1.20}$$

then

$$C_i(\phi) = \exp\{ - \sum_{j \in \partial_i^0} w_{i,j} \} + \exp\{ - \sum_{j \in \partial_i^1} w_{i,j} \}. \tag{1.21}$$

For the optimization of Eq. (1.19), an unconstrained nonlinear optimization simplex method was used [47].

1.4 Multiscale-COS/CCC Segmentation Scheme

In order to improve accuracy in the detection of text with varying size, we incorporated a multiscale framework into the COS/CCC segmentation algorithm. The multiscale framework allows us to detect both large and small components by combining results from different resolutions [48–50]. Since the COS algorithm uses a single block size (i.e. single scale), we found that large blocks are typically better suited for detecting large text, and small blocks are better suited for small text. In order to improve the detection of both large and small text, we use a multiscale segmentation scheme which uses the results of coarse-scale segmentations to guide segmentation on finer scales.

The original multiscale scheme is an approach to use coarser and finer resolution images to capture global and local contextual features. The multiscale scheme has been used for MAP segmentation in several publications [48–51]. In [48], the proposed segmentation performs ICM in a coarse-to-fine sequence, and the MAP optimization on each layer is initialized with the solution from the previous coarser resolution layer. In [49], the MRF model is replaced with a multiscale random filed (MSRF) and the MAP estimator is replaced with a sequential MAP (SMAP) estimator. The fundamental assumption of the MSRF model is that the sequence of random fields from coarse to fine scale forms a Markov chain. Therefore, the distribution of $X^{(n)}$ given all coarser scale fields is only dependent on one coarse scale field $X^{(n+1)}$. In the paper, the neighborhood beyond layers is defined in two different ways: quadtree structure model and pyramidal graph model. Also, the MAP estimator is slightly modified so that the error on the coarser scale fields is more heavily weighted than finer scale fields. In [52], the neighborhood system was further expanded to three dimension. In this model, each class label depends on class labels at both the same scale and adjacent scales. In [53], the approach uses multiscale models for both the data and the context simultaneously.

Figure 1.12 shows the overview of our multiscale-COS/CCC scheme. In the multiscale scheme, segmentation progresses from coarse to fine scales, where the coarser scales use larger block sizes, and finer scales use smaller block sizes. Each scale is numbered from $L-1$ to 0, where $L-1$ is the coarsest scale and 0 is the finest scale. Note that both COS and CCC segmentations are performed on each scale, however, only COS is adapted to the multiscale scheme. The COS algorithm is modified to use different block sizes for each scale (denoted as $m^{(n)}$), and incorporates the previous coarser segmentation result by adding a new term to the cost function.

The new cost function for the multiscale scheme is shown in Eq. (1.22). It is a function of the class assignments on both the n^{th} and $n+1^{th}$ scale.

$$f_2^{(n)}(S^{(n)}) = \sum_{i=1}^{M}\sum_{j=1}^{N}\left\{f_1^{(n)}(S^{(n)}) + \lambda_4^{(n)} V_4(s_{i,j}^{(n)}, x_{B_{i,j}}^{(n+1)})\right\} \tag{1.22}$$

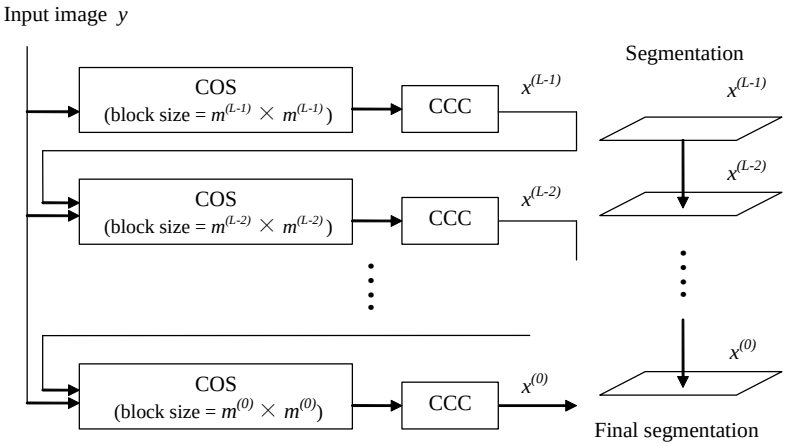


Fig. 1.12. Illustration of a multiscale-COS/CCC algorithm. Segmentation progresses from coarse to fine scales, incorporating the segmentation result from the previous coarser scale. Both COS and CCC are performed on each scale, however only COS was adapted to the multiscale scheme.

where $(\cdot)^{(n)}$ denotes the term for the n^{th} scale, and $S^{(n)}$ is the set of class assignments for the scale, that is $S = [s_{i,j}^{(n)}]$ for all i, j. The term $B_{i,j}$ is a set of pixels in a block at the position (i, j), and the term $x_{B_{i,j}}$ is the segmentation result for the block.

As it is shown in the equation, this modified cost function incorporates a new term V_4 that makes the segmentation consistent with the previous coarser scale. The term V_4, is defined as the number of mismatched pixels within the block $B_{i,j}$ between the current scale segmentation $x_{B_{i,j}}^{(n)}$ and the previous coarser scale segmentation $x_{B_{i,j}}^{(n+1)}$. The exception is that only the pixels that switch from "1" (foreground) to "0" (background) are counted when $s_{i,j}^{(n)} = 0$ or $s_{i,j}^{(n)} = 1$. This term encourages a more detailed segmentation as we proceed to finer scales. The V_4 term is normalized by dividing by the block size on the current scale. Note that the V_4 term is ignored for the coarsest scale. Using the new cost function, we find the class assignments, $s_{i,j} \in \{0, 1, 2, 3\}$, for each scale.

The parameter estimation of the cost function $f_2^{(n)}$ for $n \in \{0, \ldots L-1\}$ is performed in an off-line task. The goal is to find the optimal parameter set

$\Theta^{(n)} = \{\lambda_1^{(n)} \dots \lambda_4^{(n)}\}$ for $n \in \{0, \dots L-1\}$. To simplify the optimization process, we first performed single scale optimization to find $\Theta'^{(n)} = \{\lambda_1^{(n)} \dots \lambda_3^{(n)}\}$ for each scale. Then, we found the optimal set of $\Theta'' = \{\lambda_4^{(0)} \dots \lambda_4^{(L-2)}\}$ given the $\{\Theta'^{(0)} \dots \Theta'^{(L-1)}\}$. The error to be minimized was the number of mismatched pixels compared to ground truth segmentations, as shown in (1.5). The ω, the weighting factor of the error for false detection, was fixed to 0.5 for the multiscale-COS/CCC training process.

1.5 MRC Encoding

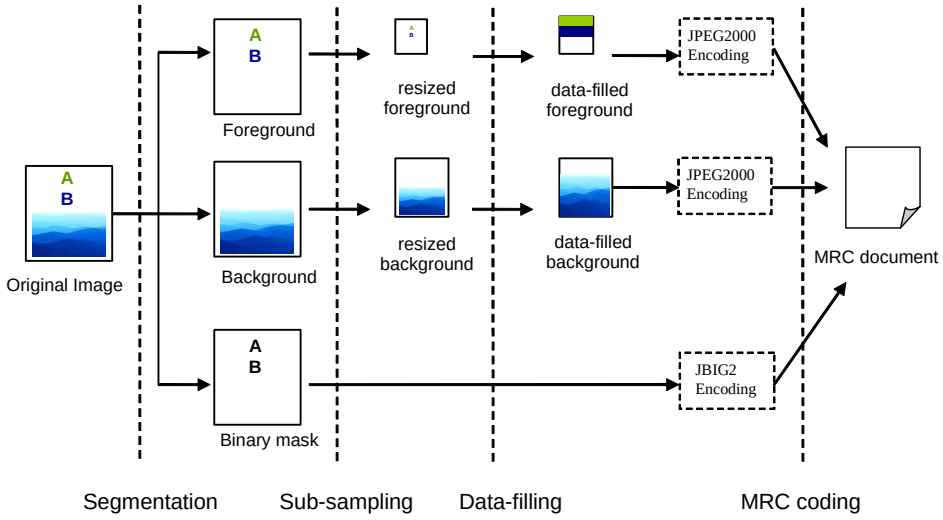


Fig. 1.13. Illustration of MRC encoding process. After an image is separated into foreground layer and background layer, each layer is subsampled to reduce the bitrate. After the subsampling, data-filling is performed to fill data to don't-care-regions. Finally, each layer is encoded and merged to create a MRC file.

MRC encoding generally consists of segmentation, subsampling, data-filling, layer compression and MRC coding. While accurate segmentation is most important to maintain the quality and reduce the bitrate of MRC documents, data-filling and subsampling are also important for efficient MRC document compression. In this chapter, the subsampling and data-filling methods which are used in this study will be explained. Although this MRC encoding procedures are not focused in this research, they are necessary to create MRC documents for the evaluation in the experimental results. Figure 1.13 shows an example of MRC encoding process. After the segmentation, an image is separated into foreground and background. Before each layer is data-filled, subsampling may be performed to reduce the bitrate. In general, higher subsampling rate is used for the foreground than the background because the foreground layer contains less spatial information than the background layer. After

the subsampling, data-filling may be performed. Finally, each layer is encoded and merged to create a MRC file. The subsampling procedure used in this research will be described in sec. 1.5.1 and the data-filling will be described in sec. 1.5.2.

1.5.1 Subsampling

Subsampling of the foreground and background layers is frequently used in MRC compression to increase the compression ratio. High ratio subsampling distorts the reconstructed image, but reduces the bit-rate dramatically. Generally, the foreground layer contains less spatial information compared to the background layer, so a higher subsampling ratio may be used for the foreground than for the background. A basic subsampling is performed by simple pixel averaging of do-care pixels in this study. (See Figure 1.14)

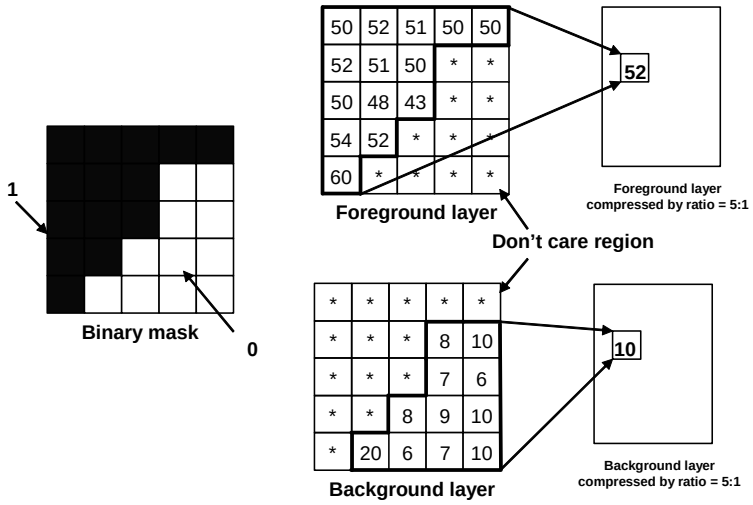


Fig. 1.14. Illustration of the subsampling used in this study. Simple pixel averaging of do-care pixels are used.

However, this basic subsampling sometimes enhances the transition values along the 0 and 1 boundaries. (See Figure 1.15) In general, the boundary between do-care regions and do-not-care regions contains many transition values. If a sub-sampling

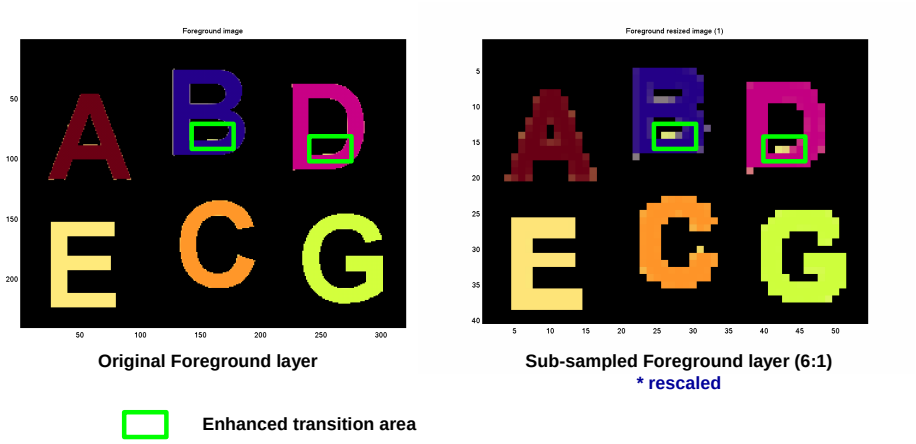


Fig. 1.15. Actual image of transition color enhancement due to subsampling. The subsampling sometimes enhances the transition values along the 0 and 1 boundaries.

block contains the only few do-care pixels along edges, the average of the transition values will be set in the reduced resolution layer.

To remove this transition enhancement, two pre-processing procedures are performed prior to the subsampling: *boundary deletion* and *8-block neighborhood averaging*. Boundary deletion is a pre-processing operation to remove unreliable pixels of do-care regions in both foreground and background layers. Boundary deletion is the same as an erosion process. For each do-care pixel, if every pixel in the pixel's 8-neighborhood is do-care pixel, the pixel remains do-care pixel, otherwise the do-care-pixel is changed to do-not-care pixel.

Eight-neighborhood averaging is another pre-processing step to remove transition values along boundaries between do-care regions "1" and do-not-care regions "0". In the blocks containing both 0's and 1's, the 0 pixels are replaced by the average of "1" pixels from the surrounding 8 blocks.

Removing transition colors and smoothing pixel values achieves higher compression ratio because the encoders applied for foreground and background layers such

as JPEG and JPEG2000 works effectively for continuous images. After these two procedures, subsampling is performed by averaging over the entire block.

1.5.2 Data filling

Data-filling is a procedure to smooth foreground and background layers by filling in areas that will ultimately be segmented out in the final decoded document. Since the foreground and the background layers are masked out by the binary layer when the image is decoded, some pixels of the foreground and background are not needed to restore the image. For example, the pixels in the foreground layer, where the corresponding pixels in the binary layer are 0, are not used in the decoder. In these "do-not-care" regions, the pixel values are arbitrary, so values may be chosen so that the subsequent compression is more efficient. JPEG, for example, produces a smaller output if the image is being compressed is more continuous.

In most cases, data-filling techniques are designed for smoothness to achieve low bitrate with a specific encoder. There are two main approaches: spatial domain data-filling and frequency domain data-filling. Region-growing methods and weighted averaging methods [54, 55] are examples of data-filling performed in the spatial domain to create visual smoothness. DCT-domain block data-filling [56] is an example of frequency domain data-filling. This approach repeats transformation and inverse-transformation with the replacement by the original do-care pixels until the smoothness converges. Similar approaches have also been proposed in [57, 58].

In this research, a combination of two spatial data-filling algorithms was used. The first data-filling strategy is a *region growing* method. In order to smooth transitions between do-care and do-not-care regions of the foreground and background layers, the do-not-care pixels are replaced by the average of 8-point neighboring do-care pixels. Later iterations use the new do-not-care pixels. Figure 1.16 shows an example of region growing based data-filling.

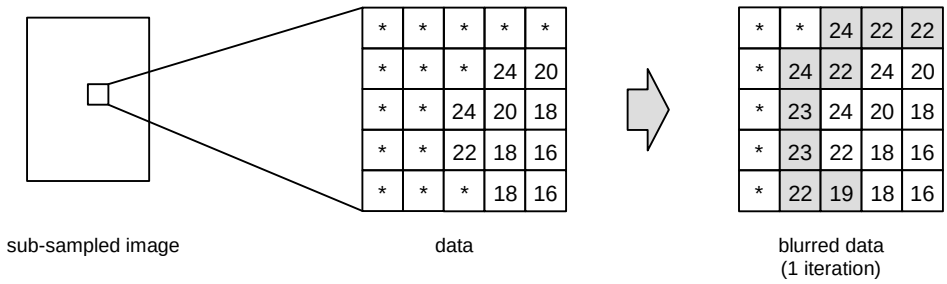


Fig. 1.16. Region growing based data-filling. In order to smooth transitions between do-care and do-not-care regions of the foreground and background layers, the do-not-care pixels are replaced by the average of 8-point neighboring do-care pixels.

Linear data-filling is another strategy which accomplishes visually smooth data-filling. This method fills do-not-care regions with linearly changing data using two sides of the regions, as shown in Eq. (1.23) and Figure 1.17. X_j is the value of the jth do-not-care pixel in the row being calculated. The pixel j_1 is the first do-care pixel to the left of the j and the pixel j_2 is the first do-care pixel to the right of j. The value of the jth pixel is computed as

$$X_j = \frac{j - j_1}{j_2 - j_1}(X_{j_2} - X_{j_1}) + X_{j_1} \tag{1.23}$$

If j_1 does not exist due to an edge of the page, X_j is filled by the value of X_{j_2}. Similarly, if j_2 does not exist due to the edge of a page, X_j is filled by the value of X_{j_1}. If neither j_1 nor j_2 exist, the entire row remains do-not-care. Vertical linear data-filling is executed after the horizontal linear data-filling in the same manner. It was possible to use this region growing method until all of the pixels are filled with values, but only 2 iterations were performed in order to limit computation. Then, the linear data-filling was applied.

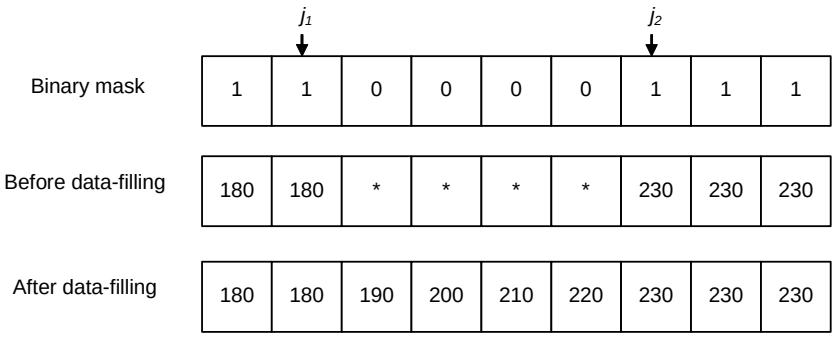


Fig. 1.17. Linear data-filling fills do-not-care regions with linearly changing data using two sides of the regions.

1.6 Results

In this section, we compare the multiscale-COS/CCC, COS/CCC, and COS segmentation results with the results of two popular thresholding methods, an MRF-based segmentation method, and two existing commercial software packages which implement MRC document compression. The thresholding methods used for the comparison in this study are Otsu [12] and Tsai [17] methods. These algorithms showed the best segmentation results among the thresholding methods in [12], [14], [15], [16], and [17]. In the actual comparison, the sRGB color image was first converted to a luma grayscale image, then each thresholding method was applied. For "Otsu/CCC" and "Tsai/CCC", the CCC algorithm was combined with the Otsu and Tsai binarization algorithms to remove false detections. In this way, we can compare the end result of the COS algorithm to alternative thresholding approaches.

The MRF-based binary segmentation used for the comparison is based on the MRF statistical model developed by Zheng and Doermann [28]. The purpose of their algorithm is to classify each component as either noise, hand-written text, or machine printed text from binary image inputs. Due to the complexity of implementation, we used a modified version of the CCC algorithm incorporating their MRF model by simply replacing our MRF classification model by their MRF noise classification model. The multiscale COS algorithm was applied without any change. The clique frequencies of their model were calculated through off-line training using a training data set. Other parameters were set as proposed in the paper.

We also used two commercial software packages for the comparison. The first package is the DjVu implementation contained in Document Express Enterprise version 5.1 [59]. DjVu is commonly used software for MRC compression and produces excellent segmentation results with efficient computation. By our observation, version 5.1 produces the best segmentation quality among the currently available DjVu packages. The second package is LuraDocument PDF Compressor, Desktop Version [26].

Both software packages extract text to create a binary mask for layered document compression.

The performance comparisons are based primarily on two aspects: the segmentation accuracy and the bitrate resulting from JBIG2 compression of the binary segmentation mask. We show samples of segmentation output and MRC decoded images using each method for a complex test image. Finally, we list the computational run times for each method.

1.6.1 Preprocessing

For consistency all scanner outputs were converted to sRGB color coordinates [60] and descreened [61] before segmentation. The scanned RGB values were first converted to an intermediate device-independent color space, CIE XYZ, then transformed to sRGB [62]. Then, Resolution Synthesis-based Denoising (RSD) [61] was applied. This descreening procedure was applied to all of the training and test images. For a fair comparison, the test images which were fed to other commercial segmentation software packages were also descreened by the same procedure.

sRGB conversion

Since different scanners usually have different RGB sensor sensitivities, light intensity and gamma correction values, it is desirable to convert scanned RGB values to standardized RGB (sRGB) values before segmentation. Converting from a device-dependent color space to a device-independent color space helps not only to obtain consistent segmentation results from different scanners, but also strengthens reproducibility.

The scanned RGB values are converted to an intermediate device-independent color space, CIE XYZ, then transformed to sRGB. The first transformation to CIE XYZ is a combination of a nonlinear transformation (NL_k) and a linear transformation (T) where $k = 0, 1$ and 2 denotes R,G and B color channels respectively [62, 63].

The nonlinear function (NL_k) may be approximated by a power-law relationship: $\phi_k(x_k) = a_k * (x_k/255)^{\gamma_k} + b_k$ where $x_k = 0, ..., 255$ is a (gamma corrected) scanned value in the k^{th} color channel and $\phi_k(x_k)$ is the corresponding ungamma corrected value. Then a 3x3 linear transformation T converts the ungamma corrected values to CIE XYZ. The second transformation from CIE XYZ to sRGB are numerically defined in the standard IEC 61966-2-1:1999. The scanner characterization model between scanned RGB and CIE XYZ is shown in Figure 1.18.

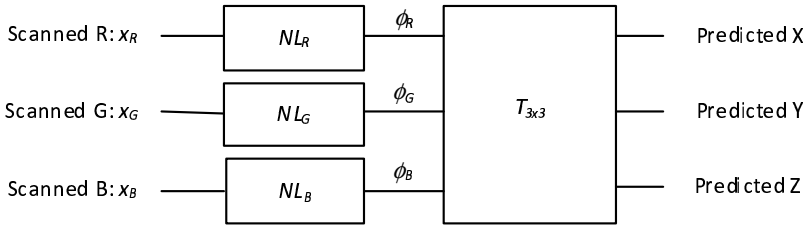


Fig. 1.18. Illustration of scanner characterization.

We used a Kodak Q.60 color target [64] for scanner color characterization. For the nonlinear transformation, we measured CIE XYZ values of the gray step colors with a spectrophotometer, then found the optimal parameters using nonlinear least square fitting. The 3x3 matrix T was then solved to minimize the error in XYZ color coordinates using the color patches on the same target.

Descreening

Most scanned images contain screening artifacts, called moiré patterns, due to the sampling of halftoned documents such as books, magazines and newspapers. The artifact level is highly dependent on scanners and the undesirable patterns sometimes cause unexpected segmentation errors. The procedure to eliminate the moiré pattern is called descreening. The simplest method of descreening is to low pass filter but this approach also softens sharp edges [61]. Desirable descreening methods supress moiré patterns while preserving sharp edges.

In this work, we used Resolution Synthesis-based Denoising (RSD) [61], which uses a modified SUSAN filter for core descreening. The SUSAN filter computes a weighted average of local pixels where the weights are determined by the spatial distance and value difference from the center pixel. The numerical settings used for the modified SUSAN filter were as follows: The filter brightness threshold, σ_b, was set to 21, and the mask size, N, was 7. The parameter σ_s embedded in the spatial filter weighting was set to 1.6. The descreening procedure was applied to all of the training and test images before segmentation was performed. For a fair comparison, the test images used in other commercial segmentation softwares were descreened by the same procedure.

1.6.2 Segmentation accuracy and bitrate

To measure the segmentation accuracy of each algorithm, we used a set of scanned documents along with corresponding "ground truth" segmentations. First, 38 documents were chosen from different document types, including flyers, newspapers, and magazines. The documents were separated into 17 training images and 21 test images, and then each document was scanned at 300 dots per inch (dpi) resolution on the Epson stylus photo RX700 scanner. After manually segmenting each of the scanned documents into text and non-text to create ground truth segmentations, we used the training images to train the algorithms, as described in the previous sections. The remaining test images were used to verify the segmentation quality. We also scanned the test documents on two additional scanners: the HP Photosmart 3300 All-in-One series and Samsung SCX-5530FN. These test images were used to examine the robustness of the algorithms to scanner variations.

The parameter values used in our results are as follows. The optimal parameter values for the multiscale-COS/CCC were shown in the Table 1.1. Three layers were used for multiscale-COS/CCC algorithm, and the block sizes were 36×36, 72×72, and 144×144. The parameters for the CCC algorithm were $p = 7.806$, $a =$

0.609, $b = 0.692$. The number of neighbors in the k-NN search was 6. In DjVu, the segmentation threshold was set to the default value while LuraDocument had no adjustable segmentation parameters.

Table 1.1
Parameter settings for the COS algorithm in multiscale-COS/CCC.

	λ_1	λ_2	λ_3	λ_4
2^{nd} layer	20.484	8.9107	17.778	1.0000
1^{st} layer	53.107	28.722	39.359	17.200
0^{th} layer	30.681	21.939	36.659	56.000

To evaluate the segmentation accuracy, we measured the percent of missed detections and false detections of segmented *components*, denoted as p_{MC} and p_{FC}. More specifically, p_{MC} and p_{FC} were computed in the following manner. The text in ground truth images were manually segmented to produce N_{gt} components. For each of these ground truth components, the corresponding location in the test segmentation was searched for a symbol of the same shape. If more than 70% of the pixels matched in the binary pattern, then the symbol was considered detected. If the total number of correctly detected components is N_d, then we define that the fraction of missed components as

$$p_{MC} = \frac{N_{gt} - N_d}{N_{gt}}. \tag{1.24}$$

Next, each correctly detected component was removed from the segmentation, and the number of remaining false components, N_{fa}, in the segmentation was counted. The fraction of false components is defined as

$$p_{FC} = \frac{N_{fa}}{N_{gt}}. \tag{1.25}$$

We also measured the percent of missed detections and false detections of individual *pixels*, denoted as p_{MP} and p_{FP}. They were computed similarly to the p_{MC} and p_{FC} defined above, except the number of pixels in the missed detections (X_{MP}) and false

detections (X_{FP}) were counted, and these numbers were then divided by the total number of pixels in the ground truth document (X_{total}).

$$p_{MP} = \frac{X_{MP}}{X_{total}}. \tag{1.26}$$

$$p_{FP} = \frac{X_{FP}}{X_{total}}. \tag{1.27}$$

Table 1.2 shows the segmentation accuracy of our algorithms (multiscale-COS/CCC, COS/CCC, and COS), the thresholding methods (Otsu and Tsai), an MRF-based algorithm (multiscale-COS/CCC/Zheng), and two commercial MRC document compression packages (DjVu and LuraDocument). The values in the Table 1.2 were calculated from all of the available test images from each scanner. Notice that multiscale-COS/CCC exhibits quite low error rate in all categories, and shows the lowest error rate in the missed component detection error p_{MC}. For the missed pixel detection p_{MP}, both multiscale-COS/CCC/Zheng and the multiscale-COS/CCC show the first and second lowest error rates. For the false detections p_{FC} and p_{FP}, the thresholding methods such as Otsu and Tsai exhibit the low error rates, however those thresholding methods show a high missed detection error rate. This is because the thresholding methods cannot separate text from background when there are multiple colors represented in the text.

For the internal comparisons among our algorithms, we observed that CCC substantially reduces the p_{FC} of the COS algorithm without increasing the p_{MC}. The multiscale-COS/CCC segmentation achieves further improvements and yields the smallest p_{MC} among our methods. Note that the missed pixel detection error rate p_{MP} in the multiscale-COS/CCC is particularly reduced compared to the other methods. This is due to the successful detection of large text components along with small text detection in the multiscale-COS/CCC. Large text influences p_{MP} more than small text since each symbol has a large number of pixels.

In the comparison of multiscale-COS/CCC to commercial products DjVu and LuraDocument, the multiscale-COS/CCC exhibits a smaller missed detection error

rate in all categories. The difference is most prominent in the false detection error rates (p_{FC}, p_{FP}).

Table 1.2
Segmentation accuracy comparison between our algorithms (multiscale-COS/CCC, COS/CCC, and COS), thresholding algorithms (Otsu and Tsai), an MRF-based algorithm (multiscale-COS/CCC/Zheng), and two commercial MRC document compression packages (DjVu and LuraDocument). Missed component error, p_{MC}, the corresponding missed pixel error, p_{MP}, false component error, p_{FC}, and the corresponding false pixel error, p_{FP}, are calculated for EPSON, HP, and Samsung scanner output.

EPSON	Multi-COS/CCC	Multi-COS/CCC/Zheng	DjVu	LuraDoc
p_{MC}	0.41%	0.95%	0.49%	4.64%
p_{MP}	0.33%	0.27%	0.47%	0.75%
p_{FC}	9.14%	9.79%	12.1%	19.5%
p_{FP}	0.45%	0.54%	1.05%	6.64%
HP	Multi-COS/CCC	Multi-COS/CCC/Zheng	DjVu	LuraDoc
p_{MC}	0.35%	1.67%	0.56%	4.84%
p_{MP}	0.20%	0.28%	0.48%	0.68%
p_{FC}	16.9%	16.8%	19.4%	41.7%
p_{FP}	0.70%	0.66%	1.19%	6.33%
Samsung	Multi-COS/CCC	Multi-COS/CCC/Zheng	DjVu	LuraDoc
p_{MC}	0.44%	1.51%	0.61%	4.50%
p_{MP}	0.32%	0.35%	0.44%	0.68%
p_{FC}	7.95%	8.10%	11.4%	19.4%
p_{FP}	0.50%	0.51%	0.81%	5.75%

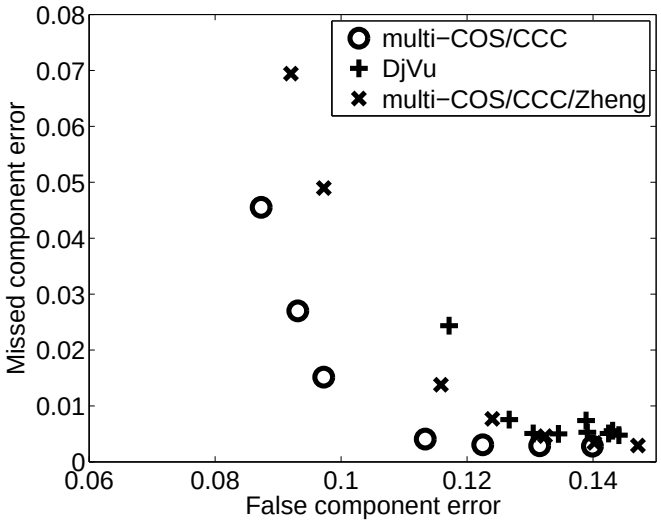
EPSON	COS/CCC	Otsu/CCC	Tsai/CCC	COS	Otsu	Tsai
p_{MC}	0.60%	2.71%	3.29%	0.53 %	4.47%	4.84%
p_{MP}	0.57%	0.55%	0.61%	0.48 %	0.64%	0.65%
p_{FC}	9.10%	9.66%	8.70%	20.1 %	25.3%	40.7%
p_{FP}	0.44%	1.60%	1.20%	3.28 %	20.4%	19.9%
HP	COS/CCC	Otsu/CCC	Tsai/CCC	COS	Otsu	Tsai
p_{MC}	0.44%	3.29%	4.17%	0.47%	4.94%	5.07%
p_{MP}	0.51%	0.57%	0.61%	0.43%	0.59%	0.60%
p_{FC}	16.9%	21.4%	19.5%	45.2%	91.6%	141.2%
p_{FP}	0.62%	1.35%	1.18%	3.04%	16.4%	15.2 %
Samsung	COS/CCC	Otsu/CCC	Tsai/CCC	COS	Otsu	Tsai
p_{MC}	0.48%	3.05%	8.59%	0.51%	5.01%	8.15 %
p_{MP}	0.53%	0.63%	0.78%	0.48%	0.66%	0.73 %
p_{FC}	7.95%	7.31%	6.67%	17.5%	20.2%	36.1 %
p_{FP}	0.33%	1.16%	0.67%	2.76%	19.0%	17.6 %

Figure 1.19 shows the trade-off between missed detection and false detection, p_{MC} vs. p_{FC} and p_{MP} vs. p_{FP}, for the multiscale-COS/CCC, multiscale-COS/CCC/Zheng, and DjVu. All three methods employ a statistical model such as an MRF or HMM for text detection. In DjVu, the trade-off between missed detection and false detection was controlled by adjustment of sensitivity levels. In multiscale-COS/CCC and multiscale-COS/CCC/Zheng method, the trade-off was controlled by the value of c_{text} in (1.18), and the c_{text} was adjusted over the interval $[-2, 5]$ for the finest layer. The results of Figure 1.19 indicate that the MRF model used by CCC results in more accurate classification of text. This is perhaps not surprising since the CCC model incorporates additional information by using component features to determine the MRF clique weights of Eq. (1.15).

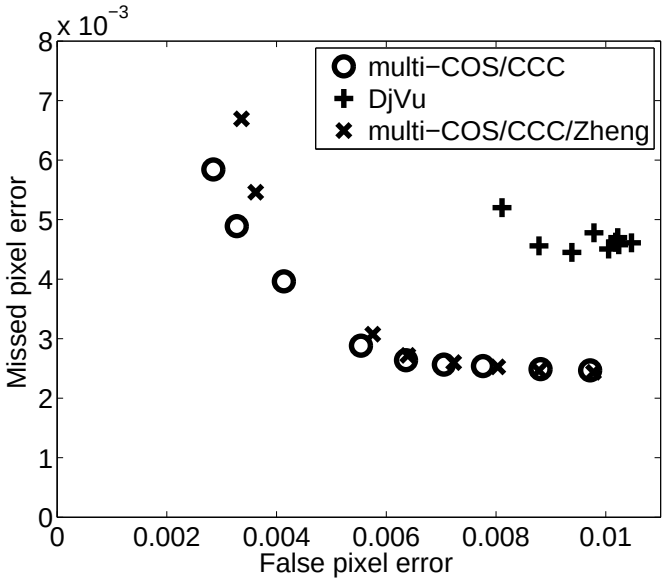
We also compared the bitrate after compression of the binary mask layer generated by multiscale-COS/CCC, multiscale-COS/CCC/Zheng, DjVu, and LuraDocument in Table 1.3. For the binary compression, we used JBIG2 [65, 66] encoding as implemented in the SnowBatch JBIG2 encoder, developed by Snowbound Software[2], using the default settings. JBIG2 is a symbol-matching based compression algorithm that works particularly well for documents containing repeated symbols such as text. Moreover, JBIG2 binary image coder generally produces the best results when used in MRC document compression. Typically, if more components are detected in a binary mask, the bitrate after compression increases. However, in the case of JBIG2, if only text components are detected in a binary mask, then the bitrate does not increase significantly because JBIG2 can store similar symbols efficiently.

Table 1.3 lists the sample mean and standard deviation (STD) of the bitrates (in bits per pixel) of multiscale-COS/CCC, multiscale-COS/CCC/Zheng, DjVu, LuraDocument, Otsu/CCC, and Tsai/CCC after compression. Notice that the bitrates of our proposed multiscale-COS/CCC method are similar or lower than DjVu, and substantially lower than LuraDocument, even though the multiscale-COS/CCC algorithm detects more text. This is likely due to the fact that the multiscale-COS/CCC

[2]http://www.snowbound.com/



(a) Trade-off between p_{MC} and p_{FC}



(b) Trade-off between p_{MP} and p_{FP}

Fig. 1.19. Comparison of multiscale-COS/CCC, multiscale-COS/CCC/Zheng, and DjVu in trade-off between missed detection error vs. false detection error. (a) component-wise (b) pixel-wise

segmentation has fewer false components than the other algorithms, thereby reducing the number of symbols to be encoded. The bitrates of the multiscale-COS/CCC

and multiscale-COS/CCC/Zheng methods are very similar while The bitrates of the Otsu/CCC and Tsai/CCC are low because many text components are missing in the binary mask.

Table 1.3
Comparison of bitrate between multiscale-COS/CCC, multiscale-COS/CCC/Zheng, DjVu, LuraDocument, Otsu/CCC, and Tsai/CCC for JBIG2 compressed binary mask layer for images scanned on EPSON, HP, and Samsung scanners.

	Multi-COS/CCC		Multi-COS/CCC/Zheng		DjVu		LuraDoc	
	average	STD	average	STD	average	STD	average	STD
EPSON (bits/pxl)	0.037	0.014	0.037	0.014	0.040	0.014	0.046	0.016
HP (bits/pxl)	0.040	0.015	0.040	0.015	0.041	0.015	0.052	0.019
Samsung (bits/pxl)	0.035	0.015	0.035	0.015	0.036	0.015	0.041	0.016

	Otsu/CCC		Tsai/CCC		ground truth	
	average	STD	average	STD	average	STD
EPSON (bits/pxl)	0.037	0.016	0.036	0.016	0.037	0.014
HP (bits/pxl)	0.040	0.016	0.040	0.016	0.039	0.016
Samsung (bits/pxl)	0.035	0.016	0.034	0.017	0.036	0.016

1.6.3 Computation time

Table 1.4 shows the computation time in seconds for multiscale-COS/CCC with 3 layers, multiscale-COS/CCC with 2 layers, COS/CCC, COS, and multiscale-COS/CCC/Zheng. We evaluated the computation time using an Intel Xeon CPU (3.20GHz), and the numbers are averaged on 21 test images. The block size on the finest resolution layer is set to 32. Notice that the computation time of multiscale segmentation grows almost linearly as the number of layers increases. The computation time of our multiscale-COS/CCC and multiscale-COS/CCC/Zheng are almost same. We also found that the computation time for Otsu and Tsai thresholding methods are 0.02 seconds for all of the test images.

Table 1.4
Computation time of multiscale-COS/CCC algorithms with 3 layers, 2 layers, COS-CCC, COS, and Multiscale-COS/CCC/Zheng.

	Multi-COS/CCC		COS/CCC	COS	Multi-COS/CCC/Zheng
	3 layers	2 layers			3 layers
Average	23.89 sec	16.32 sec	8.73 sec	5.39 sec	23.91 sec
STD	3.16 sec	2.12 sec	1.15 sec	0.39 sec	3.19 sec

1.6.4 Qualitative results

Figure 1.21 and Figure 1.22 illustrates segmentations generated by Otsu/CCC, DjVu, LuraDocument, COS, COS/CCC, multiscale-COS/CCC, and multiscale-COS/CCC/Zheng for a 300 dpi test image. The original image and ground truth segmentation are also shown. This test image contains many complex features such as different color text, light-color text on a dark background, and various sizes of text. As it is shown, COS accurately detects most text components but the number of false detections is quite large. However, COS/CCC eliminates most of these false detections without significantly sacrificing text detection. In addition, multiscale-COS/CCC generally detects both large and small text with minimal false component detection. Otsu/CCC method misses many text detections. LuraDocument is very sensitive to sharp edges embedded in picture regions and detects a large number of false components. DjVu also detects some false components but the error is less severe than LuraDocument. Multiscale-COS/CCC/Zheng's result is similar to our multiscale-COS/CCC result but our text detection error is slightly less.

Figure 1.23, Figure 1.24, Figure 1.25, and Figure 1.26 show a close up of text regions and picture regions from the same test image. In the text regions, our algorithms (COS, COS/CCC, and multiscale-COS/CCC), multiscale-COS/CCC/Zheng, and Otsu/CCC provided detailed text detection while DjVu and LuraDocument missed sections of these text components. In the picture regions, while our COS

algorithm contains many false detections, COS/CCC and multiscale-COS/CCC algorithms are much less susceptible to these false detections. The false detections by COS/CCC and multiscale-COS/CCC are also less than DjVu, LuraDocument, and multiscale-COS/CCC/Zheng.

Figure 1.27, Figure 1.28, Figure 1.29, and Figure 1.30 show MRC decoded images when the encodings relied on segmentations from Ground truth, Otsu/CCC, DjVu, LuraDocument, COS, COS/CCC, multiscale-COS/CCC, and multiscale-COS/CCC/Zheng. The examples from text and picture regions illustrate how segmentation accuracy affects the decoded image quality. Note that the MRC encoding method used after segmentation is different for each package, and MRC encoders used in DjVu and LuraDocument are not open source, therefore we developed our own MRC encoding scheme. This comparison is not strictly limited to segmentation effects, but it provides an illustration of how missed components and false component detection affects the decoded images.

As shown in Figure 1.27 and Figure 1.28, all of the text from COS, COS/CCC, multiscale-COS/CCC, and multiscale-COS/CCC/Zheng is clearly represented. Some text in the decoded images from Otsu/CCC, DjVu, and LuraDocument are blurred because missed detection placed these components in the background. In the picture region, our methods classify most of the parts as background so there is little visible distortion due to mis-segmentation. On the other hand, the falsely detected components in DjVu and LuraDocument generate artifacts in the decoded images. This is because the text-detected regions are represented in the foreground layer, therefore the image in those locations is encoded at a much lower spatial resolution.

1.6.5 Prior model evaluation

In this section, we will evaluate our selected prior model. We used the initial segmentation result generated by COS with a single block size 32×32. Then we performed the CCC segmentation with the same parameter set described in the pre-

vious section. Figure 1.31 shows the local conditional probability of each connected component given its neighbors' classes for two test images. The colored components indicate the foreground regions segmented by the COS algorithm. The yellowish or redish components were classified as text by the CCC algorithm, whereas the bluish components were classified as non-text. The brightness of each connected component indicates the intensity of the conditional probability which is described as $P(x_i|x_{\partial i})$. As shown, the conditional probability of assigned classification are close to 1 for most components. We observed that the components on boundaries between text and non-text regions take slightly smaller values but overall this local conditional probability map shows that the contextual model fits the test data well, and that the prior term contributes to an accurate classification.

We also compared the prior models with different augmented feature vector selections. We used a modified Akaike Information Criterion (AIC) to measure the goodness for model fitting [67]. The prior model evaluation criteria (referred as entropy) used in this study is defined as follows.

$$\hat{H}_{AIC} = -\frac{1}{N}\sum_{i=1}^{N} \log P(x_i|x_{\partial i}, \hat{\theta}_{ML}) + \frac{\kappa}{N}. \tag{1.28}$$

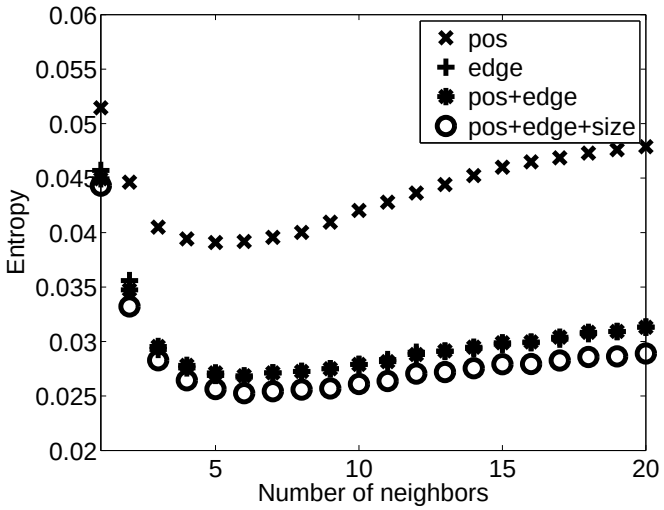
where the term κ is the number of parameters in the model. A small value of the evaluation criteria indicates a good fit. Note that loglikelihood was replaced with pseudologlikelihood in our study.

We calculated entropy for four different types of D-dimensional augmented feature vectors z:

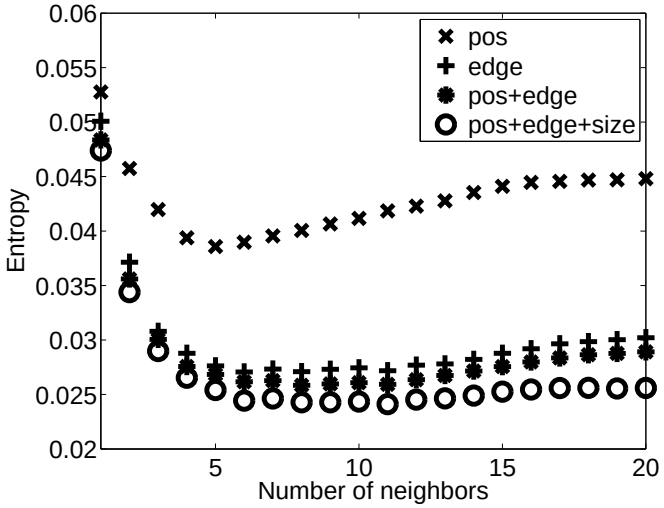
1. Geometrical information ($D = 2$): $z_i = [a_{1i}\ a_{2i}]^T$
2. Edge information ($D = 4$): $z_i = [y_{1i}\ y_{2i}\ y_{3i}\ y_{4i}\ ]^T$
3. Edge + geometrical position ($D = 6$) : $z_i = [y_{1i}\ y_{2i}\ y_{3i}\ y_{4i}\ a_{1i}\ a_{2i}]^T$
4. Edge + geometrical + size information ($D = 7$): $z_i = [y_{1i}\ y_{2i}\ y_{3i}\ y_{4i}\ a_{1i}\ a_{2i}\ s_i]^T$

where a_{1i} and a_{2i} are the geometrical position of the i^{th} connected component, and s_i is the size in pixels. The edge information $z_i = [y_{1i}\ y_{2i}\ y_{3i}\ y_{4i}]$ is the default feature vector used in the CCC algorithm. The details of the augmented feature vector is described in Appendix A.

Figure 1.20 shows the calculated entropy versus number of neighbors, k, for a training image set and a test image set. As it is shown in the result of a training set, the entropies are minimum at around $k = 6$ in most cases. If the feature vector contains only geometrical position information, the curve is sharper and slightly shifted upward. Overall, the feature vector that contains edge, position, and size information yields the smallest entropy. However, we found that the 7-D augmented feature vector does not always generate small entropy. For example, the entropy of large headline text becomes relatively large because it is not surrounded by many similar sized components. Note that large headline text does not appear often in terms of number of components, but its accurate segmentation is critical. In our study, we chose a 6-D augmented feature vector for this reason.



(a) Training set

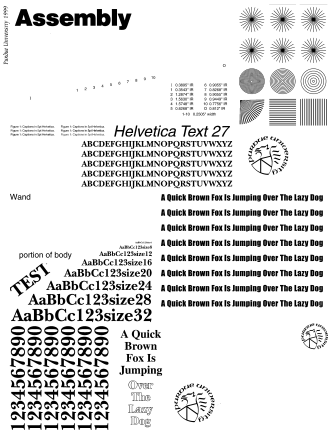


(b) Test set

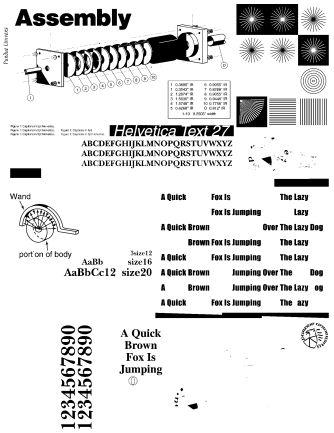
Fig. 1.20. Plot of AIC prior evaluation criteria vs. number of neighbors. The plot shows the results of four different priors with different dimension of augmented feature vectors. Small values indicate a good model fit.



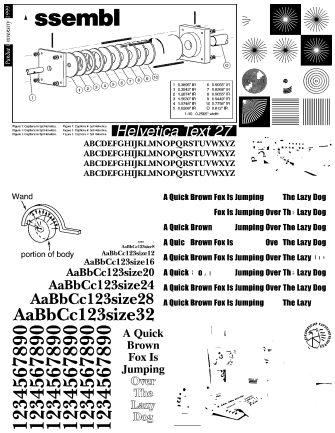
(a) Original



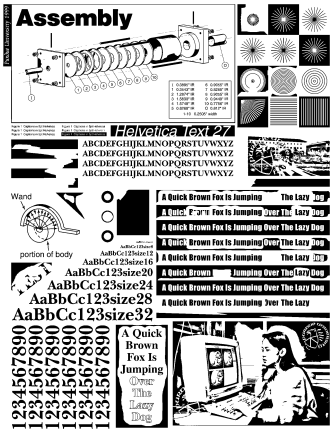
(b) Ground Truth



(c) Otsu/CCC

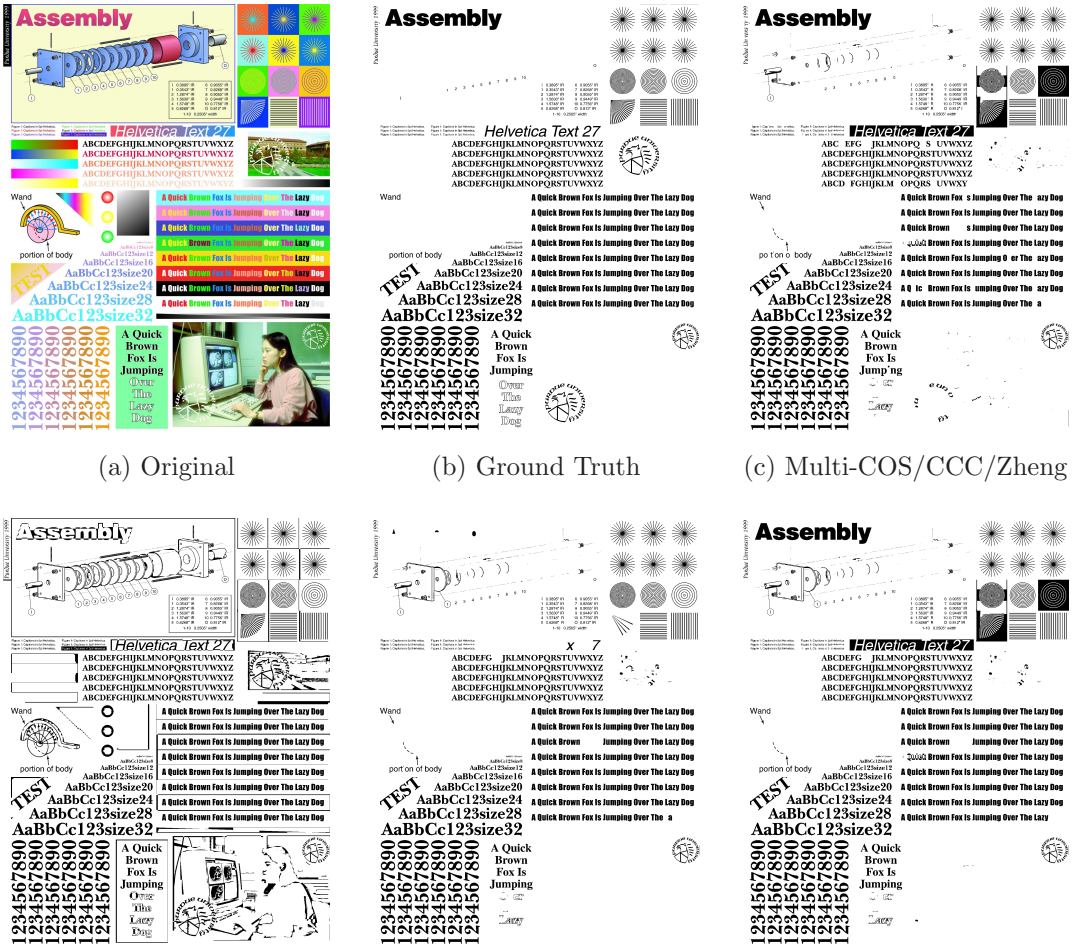


(d) DjVu



(e) LuraDocument

Fig. 1.21. Binary masks generated from Otsu/CCC, DjVu, LuraDocument. (a) Original test image (b) Ground truth segmentation (c) Otsu/CCC (d) DjVu (e) LuraDocument



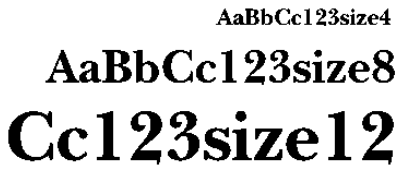
(a) Original (b) Ground Truth (c) Multi-COS/CCC/Zheng

(d) COS (e) COS/CCC (f) multiscale-COS/CCC

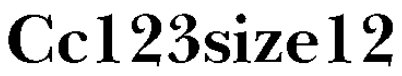
Fig. 1.22. Binary masks generated from multiscale-COS/CCC/Zheng, COS, COS/CCC, and multiscale-COS/CCC. (a) Original test image (b) Ground truth segmentation (c) Multiscale-COS/CCC/Zheng (d) COS (e) COS/CCC (f) Multiscale-COS/CCC



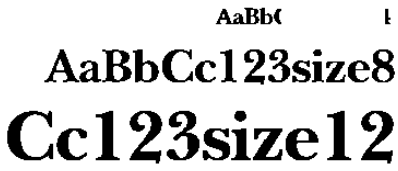
(a) Original



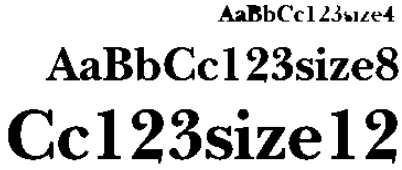
(b) Ground truth



(c) Otsu/CCC



(d) DjVu

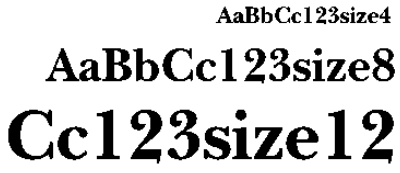


(e) LuraDocument

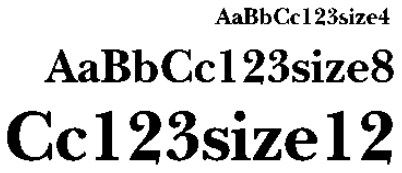
Fig. 1.23. Text regions in the binary mask. The region is 165 × 370 pixels at 400 dpi, which corresponds to 1.04 cm × 2.34 cm. (a) Original test image (b) Ground truth segmentation (c) Otsu/CCC (d) DjVu (e) LuraDocument



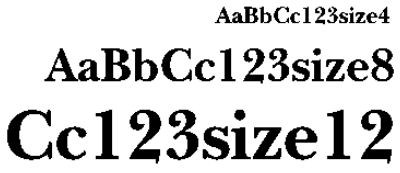
(a) Original



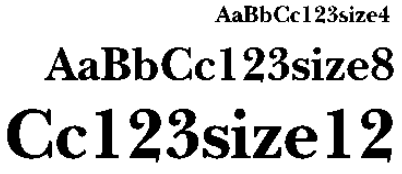
(b) Ground truth



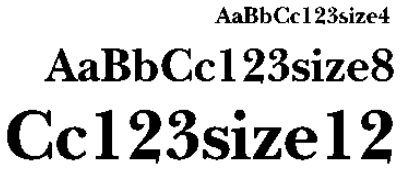
(c) Multi-COS/CCC/Zheng



(d) COS



(e) COS/CCC



(f) multiscale-COS/CCC

Fig. 1.24. Text regions in the binary mask. The region is 165 × 370 pixels at 400 dpi, which corresponds to 1.04 cm × 2.34 cm. (a) Original test image (b) Ground truth segmentation (c) Multiscale-COS/CCC/Zheng (d) COS (e) COS/CCC (f) Multiscale-COS/CCC



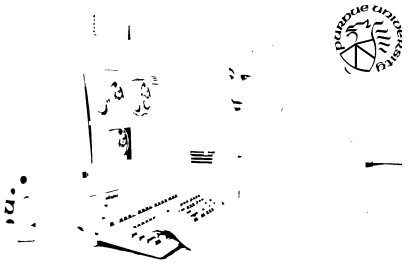
(a) Original



(b) Ground truth



(c) Otsu/CCC



(d) DjVu



(e) LuraDocument

Fig. 1.25. Picture regions in the binary mask. Picture region is 1516 × 1003 pixels at 400 dpi, which corresponds to 9.63 cm × 6.35 cm. (a) Original test image (b) Ground truth segmentation (c) Otsu/CCC (d) DjVu (e) LuraDocument



(a) Original



(b) Ground truth



(c) Multi-COS/CCC/Zheng



(d) COS



(e) COS/CCC



(f) multiscale-COS/CCC

Fig. 1.26. Picture regions in the binary mask. Picture region is 1516 × 1003 pixels at 400 dpi, which corresponds to 9.63 cm × 6.35 cm. (a) Original test image (b) Ground truth segmentation (c) Multiscale-COS/CCC/Zheng (d) COS (e) COS/CCC (f) Multiscale-COS/CCC



(a) Original



(b) Ground Truth



(c) Otsu/CCC



(d) DjVu



(e) LuraDocument

Fig. 1.27. Decoded MRC image of text regions (400 dpi). (a) Original test image (b) Ground truth (300:1 compression) (c) Otsu/CCC (311:1) (e) DjVu (281:1) (f) LuraDocument (242:1)



(a) Original

(b) Ground Truth

(c) Multiscale-COS/CCC/Zheng

(d) COS

(e) COS/CCC

(f) multiscale-COS/CCC

Fig. 1.28. Decoded MRC image of text regions (400 dpi). (a) Original test image (b) Ground truth (300:1 compression) (c) Multiscale-COS/CCC/Zheng (295:1) (d) COS (244:1) (e) COS/CCC (300:1) (f) Multiscale-COS/CCC (289:1).



(a) Original



(b) Ground Truth



(c) Otsu/CCC



(d) DjVu



(e) LuraDocument

Fig. 1.29. Decoded MRC image of picture regions (400 dpi). (a) Original test image (b) Ground truth (300:1 compression) (c) Otsu/CCC (311:1) (d) DjVu (281:1) (e) LuraDocument (242:1)



(a) Original



(b) Ground Truth



(c) Multi-COS/CCC/Zheng



(d) COS



(e) COS/CCC



(f) multiscale-COS/CCC

Fig. 1.30. Decoded MRC image of picture regions (400 dpi). (a) Original test image (b) Ground truth (300:1 compression) (c) Multiscale-COS/CCC/Zheng (295:1) (d) COS (244:1) (e) COS/CCC (300:1) (f) Multiscale-COS/CCC (289:1).



(a) An original training image (b) Local probabilities



(d) An original test image (e) Local probabilities

Fig. 1.31. The yellowish or redish components were classified as text by the CCC algorithm, whereas the bluish components were classified as non-text. The brightness of each connected component indicates the intensity of the conditional probability which is described as $P(x_i|x_{\partial i})$.

1.7 Summary

We presented a novel segmentation algorithm for the compression of raster documents. While the COS algorithm generates consistent initial segmentations, the CCC algorithm substantially reduces false detections through the use of a component-wise MRF context model. The MRF model uses a pair-wise Gibbs distribution which more heavily weights nearby components with similar features. We showed that the multiscale-COS/CCC algorithm achieves greater text detection accuracy with a lower false detection rate, as compared to state-of-the-art commercial MRC products. Such text-only segmentations are also potentially useful for document processing applications such as OCR.

Although our segmentation algorithms show promising results, there are some minor situations for which we could possibly improve the segmentation quality. First, if text does not have two distinct colors across the edge (e.g. shadow), the COS algorithm sometimes causes unexpected mis-segmentation. This is because our algorithm assumed text and background must have two distinct peaks in the histogram. Serious moiré pattern on scanned images can also possibly cause mis-segmentation. In our study, we applied RSD descreening to eliminate the artifact in the preprocessing procedure. However, if a more suitable descreening procedure is developed, then better segmentation accuracy might be obtained.

2. AUTOMATIC CONTRAST ENHANCEMENT SCHEME FOR A DIGITAL SCANNED IMAGE

2.1 Introduction

The motivation for this study is that the background of scanned images by home scanners can sometimes appear too dark. This is particularly true for scans of paper materials such as newspapers, magazines and phone books. In fact, more accurate the measured RGB tristimulus values are, the perceived color sometimes appears darker.

Figure 2.1 shows examples of a newspaper scanned at 300 dpi by Epson stylus photo RX700, HP Photosmart 3300 All-in-One series, and Samsung SCX-5530FN. Notice that the scanned image by the Samsung SCX-5530FN scanner is similar in appearance to a newspaper, but the background is too dark to read the contents. A lighter and more uniform background color is typically preferred because it has the advantages that the contrast of the image is more distinct, the background noise is reduced, and the background color is more consistent across paper materials.

Based on this motivation, our goal of this project is to develop a robust algorithm for automatic contrast enhancement. More specifically, our objective is to snap the background color of the paper material to display-white and the full black colorant of the printer to display black without large color shift. In addition, we want to reduce show-through effects of thin paper materials. Thin paper materials such as typical of phonebooks and magazines sometimes allow the images on the backside of the paper to be visible from the front page. Therefore, it is desirable to block the show-through effect and make the background clean [68].

Our automatic color enhancement algorithm described in this document consists of four components: auto cropping, paper-white estimation, paper-black estimation, and linear contrast stretch. First, the algorithm performs auto cropping of an input



Fig. 2.1. Scanned image examples of a newspaper

image to extract the region which contains only paper. This is because if extra white or black is included around the borders of the image, the estimated paper white or estimated black colorant can be wrong. Note that the cropped region can be smaller than the actual scanned document, but should not be larger than it. Since the auto cropping is for accurate paper white estimation, excessive cropping is allowed as long as the cropped region contains the paper color.

Next, paper-white color and black colorant color are estimated using the maximum and minimum values for each RGB channel. If the estimated paper-white and black colorant is close enough to neutral white and neutral black, we perform contrast stretching. If not, we do not perform contrast stretch to avoid large color shift. The contrast stretch is a simple linear stretching by snapping the estimated paper-white and estimated black colorant to the largest and smallest encoded values. In addition, our method can prevent show-through of thin paper materials by aggressively snapping paper-white to the largest encoded value.

Much research has been done for automatic contrast enhancement in several areas. For the paper-white estimation, similar research has been done for illuminant estimation of digital camera and video [69]. Illuminant estimation is used for automatic

white balance so as to convert an image under unknown illumination to an image under known illumination.

Probably the most famous algorithms for automatic white balance are the grayworld algorithm [70] and the scale-by-max algorithm [69]. The grayworld algorithm estimates the current illumination by the average of the entire captured image, based on the assumption that if there is a wide distribution of colors in an image, the average reflected color should be the color of the light. The disadvantage of the grayworld algorithm is the converted images tend to appear bright. The scale-by-max algorithm estimates the unknown illumination by the maximum response in each color channel. The disadvantage of the scale-by-max algorithm is that if an image contains bright pixels locally, the contrast stretch does not work well. There are other sophisticated methods such as the gamut mapping method [71,72] and color correlation method [73] but they are not suitable for our purpose because these algorithms basically work by estimating illumination type.

Chromatic adaptation is one of the related research areas to contrast stretch. The term chromatic adaptation originally referred to the ability of the human visual system to discount the color of the illumination and preserve the appearance of an object. There are several schemes have been proposed to mimic the chromatic adaptation such as Von Kries adaptation, Nayatani model, Guth's model and the Fairchild model [74, 75]. Among these methods, Von Kries is the probably simplest and the most popular method. In Von Kries method, chromatic adaptation is performed as an independent gain regulation of the three sensor responses of the human visual system. Typically, the Von Kries chromatic adaptation is applied to the tristimulus values of human sensor responses, however it is sometimes directly applied to the RGB values captured by a target device without any conversion.

In general, contrast stretch algorithms can be divided into three categories: linear contrast stretch, non-linear contrast stretch and histogram equalization. For example, Von Kries method is one of the linear contrast stretch methods. Each sensor response is linearly and independently stretched. Non-linear contrast stretch uses a

non-linear transformation for the conversion from input to output. The most famous conversion is called an S-shape transformation where the graph is steep in the middle and relatively flat at the ends. Histogram equalization converts the data so that it is distributed uniformly over the full color range [76].

This chapter is organized as follows. Section 2.2 explains the auto cropping algorithm. Section 2.3 and section 2.4 describes the paper-white estimation and black colorant estimation, and section 2.5 describes the contrast stretching method. Show-through blocking is then explained in the section 2.6. Finally section 2.7 shows the results and comparison of Samsung, EPSON and HP scanners.

2.2 Auto Cropping

Auto cropping extracts the paper location on the scanner glass from a scanned image. When an image is scanned, an extra white border is sometimes included around the image because of the reflection of the scanner lid. If the lid is open, this border might be black. If the extra white or black is included, the estimated paper white or estimated black colorant can be misidentified. Therefore, auto cropping is needed for accurate paper white and black colorant estimation.

Note that the auto cropping does not need to be precise. Since our goal is paper-white and black-colorant estimation, excessive cropping is allowed as long as the cropped region contains region with the paper color. However, although the cropped region can be smaller than the actual paper region, it should not be larger than the actual paper region.

The challenging issue of auto cropping is that if the scanned paper color is close to lid-white, it is difficult to extract the paper location because the boundary is not obvious. Also, if the scanned image is tilted, the paper boundary detection is further complicated.

The auto cropping procedure presented in this document consists of two parts: region growing and convex hull check. The region growing is the initial cropping

process, and the convex hull check compensates for the errors from the initial cropping results.

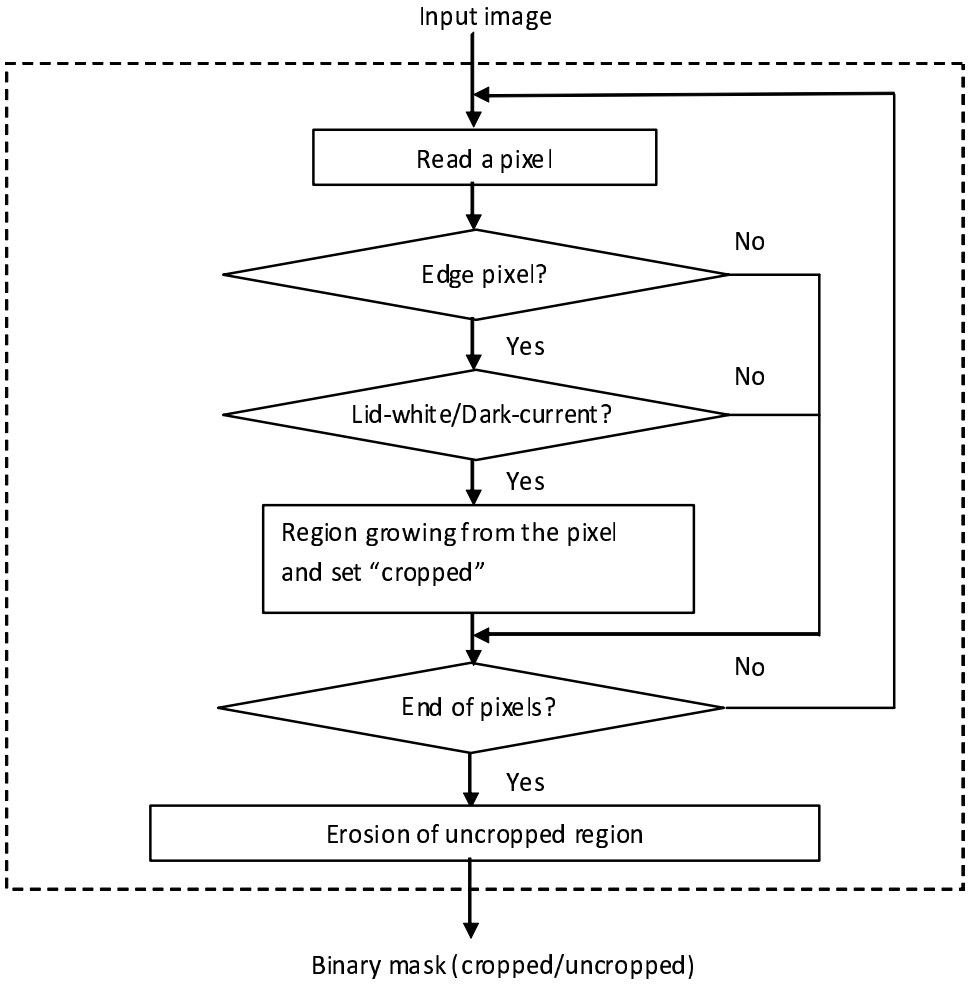


Fig. 2.2. Flowchart of region growing auto cropping

Figure 2.2 shows the flowchart of the region growing auto cropping. First, cropping is performed by region growing from the four edges of the scanned image. A single pixel is read along the edges of the image, then if the pixel color is lid-white or pure-black (dark current), region growing is performed from the pixel, and the extracted region is labeled as "cropped." This procedure is repeated until all of the edge

1. A single pixel is read along an edge of image.
   If the pixel color is lid-white, add the pixel to a "waiting list" and label the pixel as "cropped."
2. While (waiting list is not empty)
   (a) Obtain a pixel s, from the waiting list
   (b) For each pixel i in 4-point neighborhood of s,
   Add i to the waiting list if
   $(R_{wb} - R_i)^2 + (G_{wb} - G_i)^2 + (B_{wb} - B_i)^2 \leq Thres_{AC}$
   Label the pixel as "cropped."

Fig. 2.3. Region growing for auto cropping

pixels are exhausted. The procedure of region growing for lid-white is described in Figure 2.3. Note that R_{wb}, G_{wb}, and B_{wb} are predetermined RGB values of lid-white.

Next, the output binary mask in which the pixels are labeled as uncropped is eroded to remove unreliable pixels. The erosion is performed with a 3×3 window. The final output binary mask is then the region of interest for the paper-white estimation and black colorant estimation.

Figure 2.4 shows the results of auto cropping from the region growing procedure. The original #1 is a newspaper image and the original #2 is a catalog image. As shown, the auto cropping for the original #1 properly removes the white regions around the newspaper. However, for the original #2, the auto cropping removes too much of the white regions because the boundary is not obvious. To compensate for this erroneous effect, we added a convex hull check to predict the boundary from the image contents.



Fig. 2.4. Examples of region growing auto cropping

The convex hull check determines if the region growing resulted in excessive boundary region removal. If this excessive boundary region removal occurred, then the cropped regions are replaced with the 8-point convex hull shaped region. Otherwise, the cropped regions remains the same.

Figure 2.5 shows the flowchart of the convex hull cropping check. First, the algorithm extracts a convex hull (8-sided polygon) circumscribing uncropped pixels of the binary mask obtained by the region growing cropping. The convex hull is approximated as an 8-sided polygon because it reduces the computation time and complexity. Figure 2.6 shows an example of the convex hull extraction of a tilted scanned image. The 8-sided polygon is extracted by narrowing the region horizontally, vertically and along 45 degree slanted lines. The convex hull cropping does not exactly match to the boundary of the scanned paper position but this cropping is sufficient to allow estimation of paper-white and black colorant. After extracting a convex hull region, the number of pixels of the intersection between the convex hull and the "cropped" pixels is counted. The counted number is then divided by the number of pixels of the convex hull, and then this number is used to determine if an

excessive cropping occurred. If the ratio is over 20%, the algorithm determines the excessive cropping occurred and replaces the region growing cropping with a convex hull cropping. Otherwise, the algorithm uses the original region growing cropping. Figure 2.7 shows examples of the final auto-cropping results by this algorithm.

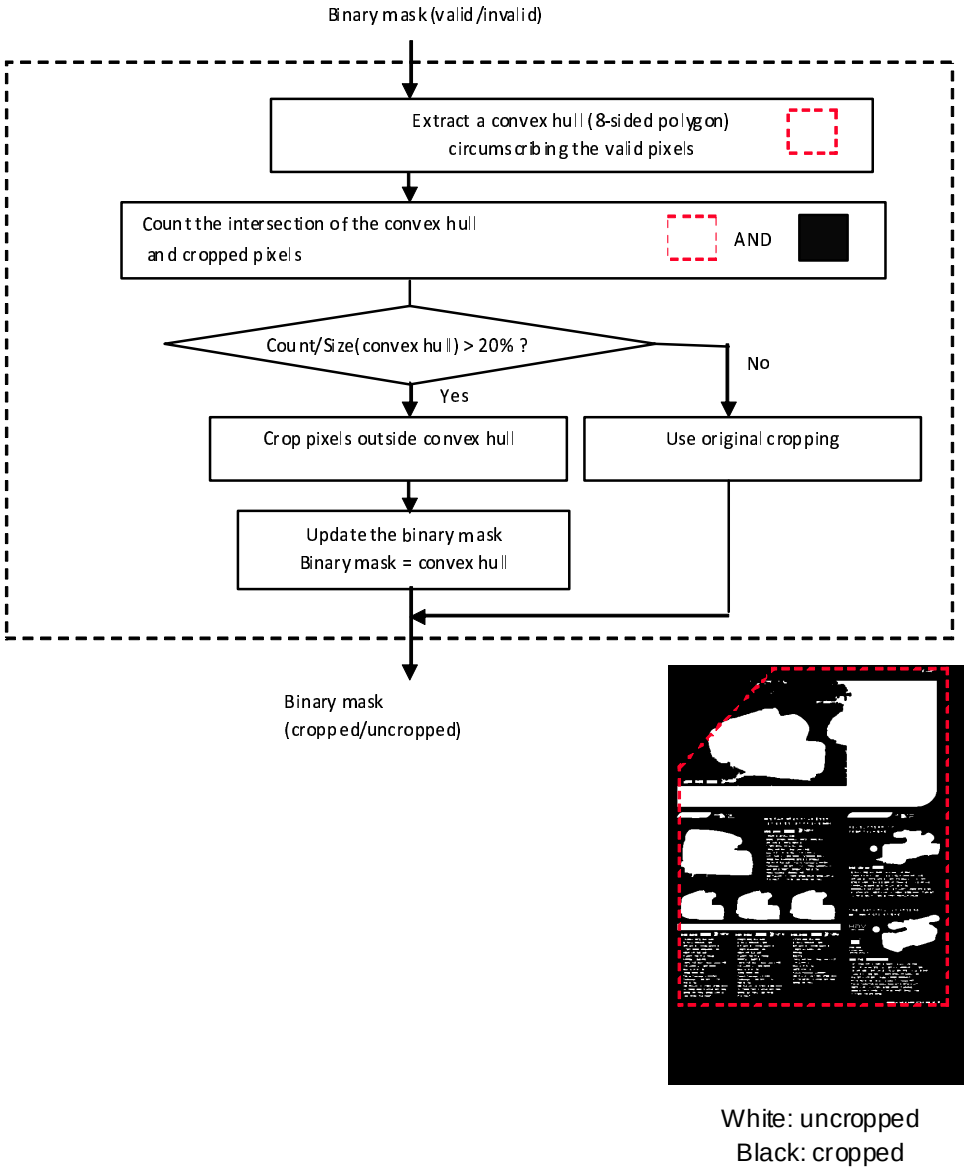


Fig. 2.5. Flowchart of convex-hull cropping

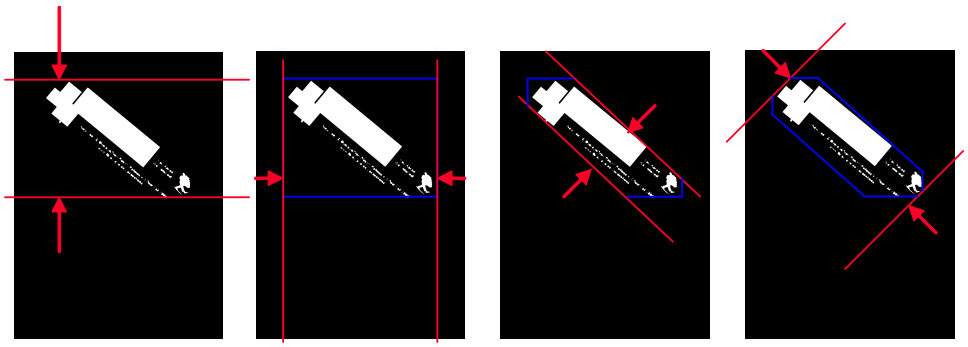


Fig. 2.6. Example of convex-hull cropping

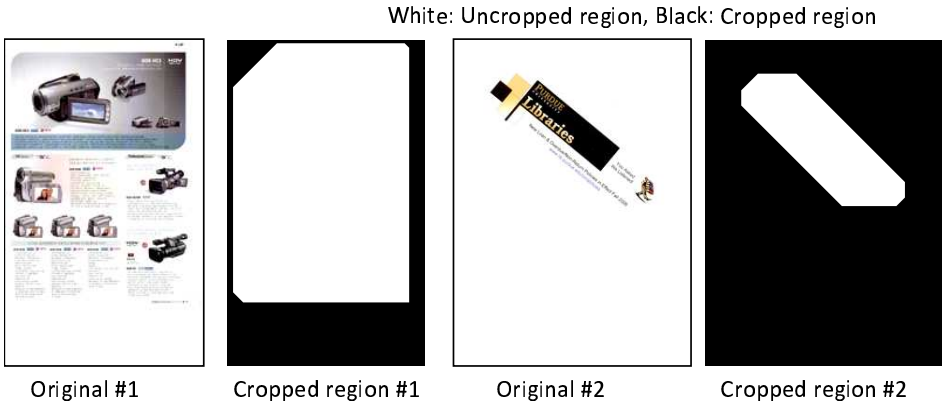


Fig. 2.7. Final result of automatic cropping

2.3 Paper-white Estimation

Paper-white estimation determines the scanned RGB value of the paper-white color from the auto cropped regions of the input image. Most techniques used for paper-white estimation employ the image histogram. For example, we can determine the paper-white color by extracting the maximum value from each RGB channel (i.e. (maxR, maxG, maxB)). However, each value of (maxR, maxG, maxB) does not necessarily come from the same pixel. Threfore, we need to find a set of pixels where

each RGB value is large simultaneously. Our approach to paper-white estimation is to 1) find the maximum value for each RGB channel, 2) perform region growing from the pixels which have the maximum value for each RGB channel, 3) take the intersection of the three binary masks generated in step-2, and 4) calculate the average RGB values for the pixels in the intersection.

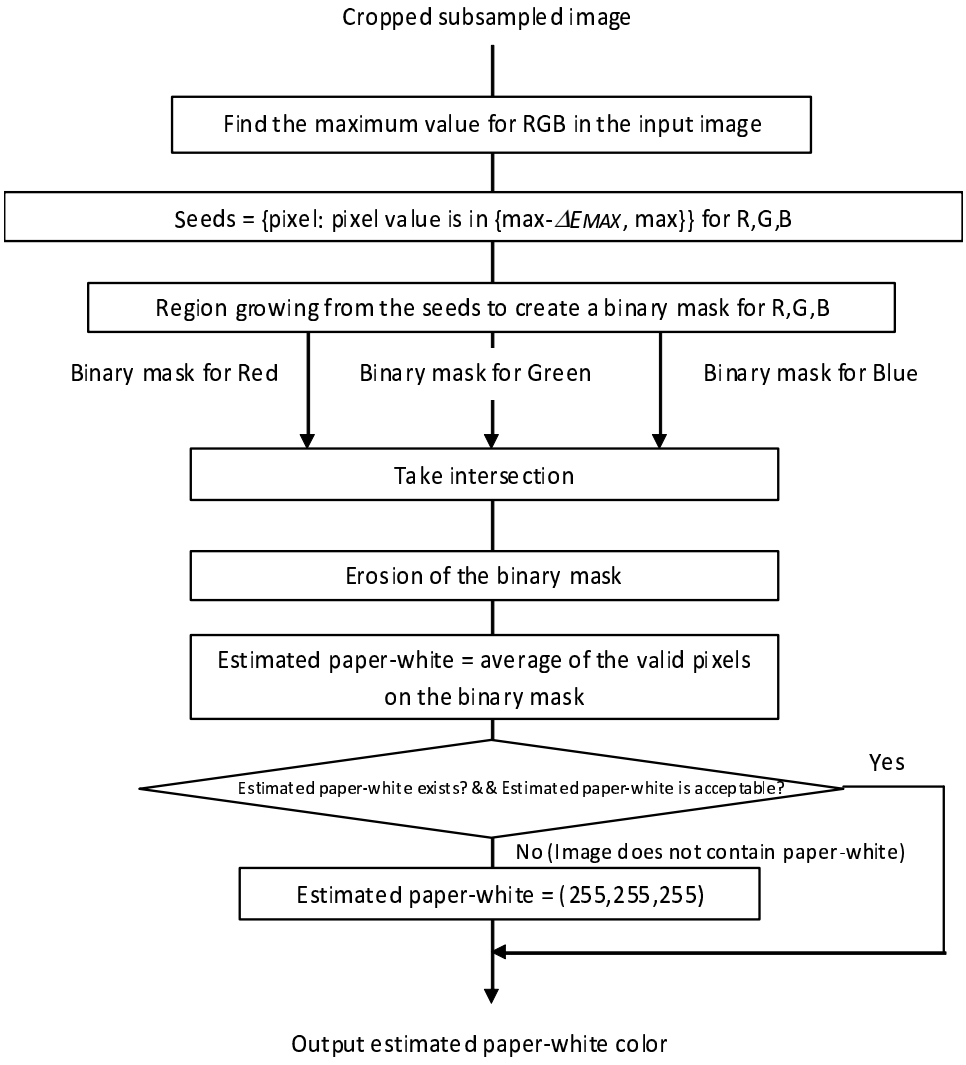


Fig. 2.8. Paper-white estimation flowchart

For each RGB channel,

1. Find pixels whose values fall in the range [max-ΔE_{MAX}, max]. Add the pixels to a set S and a waiting list W.
2. While W is not empty,
   (a) Obtain a pixel v from W
   (b) Compute average color over S: $(R_{avg}, G_{avg}, B_{avg})$
   (c) For each pixel i in a 4-point neighborhood of v,
   Add i to S and W if
   $(R_{avg} - R_i)^2 + (G_{avg} - G_i)^2 + (B_{avg} - B_i)^2 \leq Thres_{PW}$

Fig. 2.9. Region growing for paper-white estimation

Figure 2.8 shows a flowchart of our paper-white estimation. First, the resolution of the input image is reduced by block averaging and decimation to save computation time. Then, the algorithm finds the maximum value for each RGB channel in the image. Next, we perform region growing for each RGB channel from the pixels which have the maximum values. To prevent region growing from being limited due to the noise variation, we select more pixels to feed to region growing. More specifically, for each RGB channel, the pixels which fall in the range [max-ΔE_{MAX}, max] are selected as the seed pixels of region growing. A large value of ΔE_{MAX} increases the number of seed pixels fed to region growing.

The region growing method used for the paper-white estimation is slightly different from regular region growing. The current pixel value is compared with the "average" of the current set of candidate pixels for region growing. This prevents the algorithm from expanding the region into gradated color areas. The outline of the region growing procedure is described in Figure 2.9.

The region growing for each RGB channel generates three binary masks. Next, we take an intersection of the three binary masks. By doing this, we obtain possible white regions. Then, the valid pixel regions of the binary mask are eroded to remove unreliable pixels using a 3×3 window.

The estimated paper-white is finally calculated by taking the average of the RGB values in the region. However, we still need to determine whether this region is white or not because the image might not contain any white color. The criterion for determination of white is:

1. The average over the three values R, G and B is greater than $THRES^{mean}_{white}$
2. The variance of the three values R, G and B is less than $THRES^{std}_{white}$

The formula of the variance is $var = \sqrt{(R-\mu)^2 + (G-\mu)^2 + (B-\mu)^2}$, where μ is the average over the three values R, G, and B. The values of $THRES^{mean}_{white}$ and $THRES^{std}_{white}$ are 180 and 25 in our study. If the estimated paper-white is determined to be white, the estimated paper-white will be used for the contrast stretch. If the estimated paper-white is determined to be non-white, the estimated paper-white is replaced by (255,255,255), which means that we do not perform snapping.

2.4 Black Colorant Estimation

The black colorant estimation determines the RGB value of the darkest black color from the auto cropped regions of the input image. The result of black colorant estimation is used for snapping the value to pure black. Our approach of the black colorant estimation is exactly the same as the paper-white estimation. The only difference is that the pixels with minimum values are searched in each color channel R, G, and B, and then region growing is performed. Figure 2.10 shows the flowchart of our black colorant estimation.

The default value of ΔE_{MIN} is set to 10 for black colorant estimation. If the image does not contain black, we do not snap pixels to black. The criterion of determination of black is as follows.

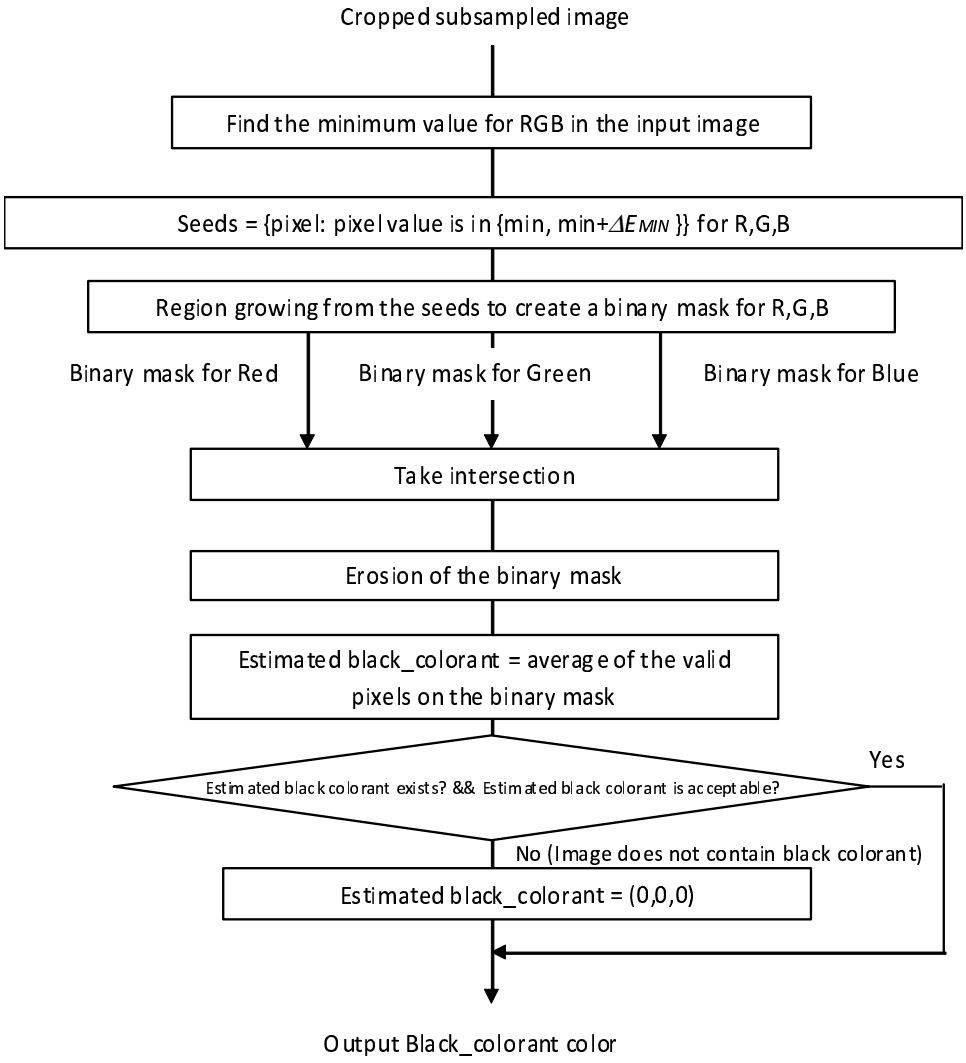


Fig. 2.10. Black colorant estimation flowchart

1. The average over the three values R, G and B is smaller than $THRES^{mean}_{black}$
2. The variance of the three values R, G and B is less than $THRES^{std}_{black}$

The formula of the variance is $var = \sqrt{(R-\mu)^2 + (G-\mu)^2 + (B-\mu)^2}$, where μ is the average over the three values R, G, and B. The default values of $THRES^{mean}_{black}$ and $THRES^{std}_{black}$ are 100 and 25 in our study. If the estimated black colorant is

determined to be black, the estimated black will be used for the contrast stretch. If the estimated black colorant is determined to be non-black, the estimated black colorant is replaced by (0,0,0), which means that we do not perform any snapping to pure black.

2.5 Contrast Stretch

In this study, independent linear contrast stretch is applied to each RGB channel for the contrast enhancement. The independent linear contrast stretch is defined as follows.

$$\begin{aligned} R_{out} &= \frac{255}{R_w - R_b}(R_{in} - R_b) \\ G_{out} &= \frac{255}{G_w - G_b}(G_{in} - G_b) \\ B_{out} &= \frac{255}{B_w - B_b}(B_{in} - B_b) \end{aligned} \tag{2.1}$$

where R_w, G_w, B_w are the estimated paper-white color, and R_b, G_b, B_b are the estimated black colorant. Linear contrast stretch is computationally efficient and does not distort the input image color severely because it preserves the local contrast.

One of the concerns of this method is color shift. Although we use the independent linear contrast stretch for RGB for this research, our proposed algorithm does not cause too much color shift because contrast stretch is performed only when estimated paper white and black colorant are neutral gray. Since each value of R_w, G_w, B_w (or R_b, G_b, B_b) are close to each other, the linear stretch for each color channel is similar.

2.6 Show-through Blocking

Typically, show-through problems occur when images printed on the backside of the paper are visible due to thin paper material. Show-through often makes the background noisy and non-uniform. The simplest strategy to prevent show-through is to snap pixels to white more aggressively. For example, we may reduce each RGB value of the estimated paper-white by $p\%$ so that more pixels are snapped to white.

Note that the optimal value of p varies for different input images. If an image has severe show-through, large p sometimes works well. However, we have to be careful because large p makes the image too bright. From our experiments, $p = 10$ works well for most of the input images.

2.7 Results

2.7.1 Paper-white estimation

In this section, we will show the quantitative error between the estimated paper-white and the ground truth paper-white. We selected 150 test images including 30 newspaper, 30 phonebook, 30 Time magazine, 30 Printed documents and 30 Wavelinks magazine published by ECE department in Purdue university. Each image is scanned by Samsung SCX-5530FN MFP scanner in 300 dpi with the default mode. The parameter settings are shown in Table 2.1.

Table 2.1
Parameter settings for automatic contrast enhancement

$Thres_{AC}$	$THRES^{mean}_{white}$	$THRES^{std}_{white}$	$THRES^{mean}_{black}$	$THRES^{std}_{black}$
30.0	180.0	25.0	100.0	25.0

The ground truth paper-white color was extracted manually for each test image. The paper-white estimation error was measured quantitatively as follows.

$$\Delta E = \sqrt{(\hat{L}_{PW} - L_{PW})^2 + (\hat{a}_{PW} - a_{PW})^2 + (\hat{b}_{PW} - b_{PW})^2} \quad (2.2)$$

$$\Delta E_L = \sqrt{(\hat{L}_{PW} - L_{PW})^2} \quad (2.3)$$

$$\Delta E_{ab} = \sqrt{(\hat{a}_{PW} - a_{PW})^2 + (\hat{b}_{PW} - b_{PW})^2} \quad (2.4)$$

where (L_{PW}, a_{PW}, b_{PW}) is the CIELAB tristimulus values of ground truth paper-white, and $(\hat{L}_{PW}, \hat{a}_{PW}, \hat{b}_{PW})$ is the CIELAB tristimulus values of estimated paper-

white. Note that we calibrated the scanner to calculate the CIELAB tristimulus values from scanned RGB values. The conversion from the scanned RGB values to CIE XYZ values are described in section 1.6.1. Then, the CIE XYZ values are numerically transformed to CIE Lab values.

Table 2.2 - Table 2.4 show the paper-white estimation error for the different region-growing thresholds in the paper white estimation: $Thres_{PW} = 10$, $Thres_{PW} = 14$, and $Thres_{PW} = 17$. The term $Thres_{PW}$ was defined in the region growing procedure of paper-white estimation section. It can be observed that if the threshold value for the region growing is large, the white estimation error is small especially for the noisy paper materials such as newspaper and phonebook.

Table 2.2
Paper-white estimation error for $Thres_{PW} = 10$

	Mean of ΔE	Mean of ΔE_L	Mean of ΔE_{ab}
Newspaper	3.2029	2.9590	0.9603
Phonebook	3.4828	3.0261	1.6042
Printed document	0.7510	0.3592	0.6025
Time magazine	2.2317	1.8576	0.9971
Wavelink magazine	1.5913	1.3777	0.5623

2.7.2 Qualitative results

In this section, we will show the qualitative image results of the automatic contrast stretch. We used Samsung SCX-5530FN scanned images to apply our algorithm. For comparison, we will also show the Epson stulus photo RX700 and HP photosmart 3300 all-in-one series scanned images, in which the contrast was already adjusted by their devices. The test images are randomly selected from various types of papers such as newspapers, magazines, and phonebooks. All of the images were scanned at

Table 2.3
Paper-white estimation error for $Thres_{PW} = 14$

	Mean of ΔE	Mean of ΔE_L	Mean of ΔE_{ab}
Newspaper	2.5881	2.3569	0.7903
Phonebook	3.3906	2.8026	1.7363
Printed document	0.8914	0.4092	0.7407
Time magazine	1.9115	1.4921	0.8521
Wavelink magazine	1.5801	1.3639	0.5648

Table 2.4
Paper-white estimation error for $Thres_{PW} = 17$

	Mean of ΔE	Mean of ΔE_L	Mean of ΔE_{ab}
Newspaper	2.3329	2.1119	0.7554
Phonebook	3.1709	2.5318	1.7051
Printed document	0.9720	0.4852	0.7951
Time magazine	1.9352	1.5297	0.9213
Wavelink magazine	1.5751	1.3576	0.5736

300 dpi by the default mode. Figure 2.11, Figure 2.12, Figure 2.13, and Figure 2.14 show the EPSON scanned image, HP scanned images, the original Samsung scanned images with no contrast adjustment, and the processed Samsung scanned images with automatic contrast adjustment. Figure 2.11 and Figure 2.12 are examples of newspapers with zoomed-in text regions, while Figure 2.13 and Figure 2.14 are examples of magazines with zoomed-in text regions.

As we can see in the figures, EPSON scanned images are darker and the color is more distinct than the other scanners. However, the white background is sometimes noisy because it is not completely snapped to the largest value. Most of the HP

scanned images have very uniform background and the white is snapped to the largest value. However, the images tend to be brighter than the other scanners. Our proposed algorithm enhanced the dynamic range of the original Samsung scanned images and the color contrast is comparable to EPSON scanner. The white background is also snapped to the largest value to be uniform.



(a) EPSON (b) HP

(c) Original Samsung (d) Adjusted Samsung

Fig. 2.11. (a) EPSON scanned image (b) HP scanned image (c) Original Samsung scanned image with no contrast adjustment (d) Processed Samsung scanned image with automatic contrast adjustment for the first example of newspaper materials.



(a) EPSON (b) HP

(c) Original Samsung (d) Adjusted Samsung

Fig. 2.12. (a) EPSON scanned image (b) HP scanned image (c) Original Samsung scanned image with no contrast adjustment (d) Processed Samsung scanned image with automatic contrast adjustment for the second example of newspaper materials.



(a) EPSON



(b) HP



(c) Original Samsung



(d) Adjusted Samsung

Fig. 2.13. (a) EPSON scanned image (b) HP scanned image (c) Original Samsung scanned image with no contrast adjustment (d) Processed Samsung scanned image with automatic contrast adjustment for the first example of magazine materials.

2.7.3 Show-through improvements

Figure 2.15 and Figure 2.16 show the results of show-through blocking. The original images have severe show-through in the background but most of them disappeared or distorted by snapping to white color.



(a) EPSON (b) HP

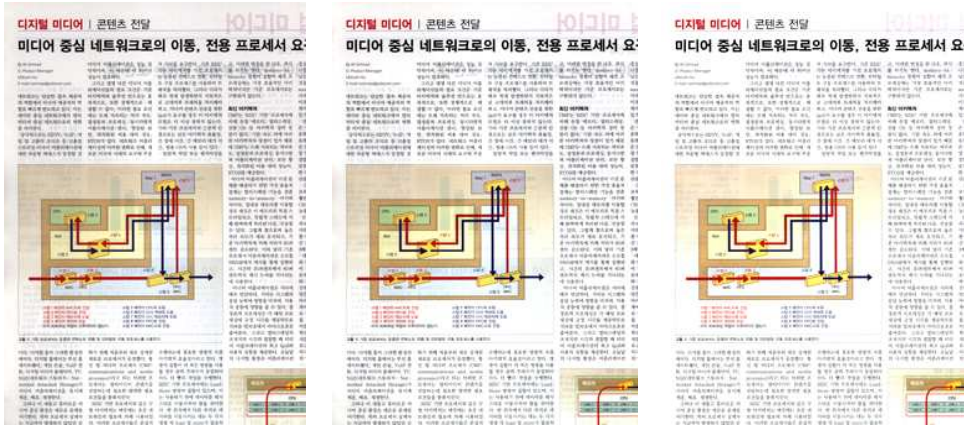
(c) Original Samsung (d) Adjusted Samsung

Fig. 2.14. (a) EPSON scanned image (b) HP scanned image (c) Original Samsung scanned image with no contrast adjustment (d) Processed Samsung scanned image with automatic contrast adjustment for the second example of magazine materials.



Original | Initial contrast stretch | Contrast stretch by reducing estimated paper-white color by 10%

Fig. 2.15. The first example of show-through blocking for Samsung scanned image.



Original | Initial contrast stretch | Contrast stretch by reducing estimated paper-white color by 10%

Fig. 2.16. The second example of show-through blocking for Samsung scanned image.

2.8 Summary

In this study, we developed a robust automatic contrast enhancement algorithm to increase the dynamic range especially for the Samsung SCX-5530FN multi functional printer (MFP) scanner. The algorithm automatically crops region of interest and estimates the paper-white and black colorant for contrast enhancement. Then the algorithm snaps the estimated paper-white color to the largest encoded value while it snaps estimated black colorant color to the smallest value without a large color shift. The algorithm can also reduce show-through effects of thin paper materials. The algorithm is robust for scanned images of any kind of paper materials, and is also computationally inexpensive.

LIST OF REFERENCES

LIST OF REFERENCES

[1] International Telecommunication Union, *ITU-T recommendation T.44 Mixed raster content (MRC)*. April 1999.

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

[3] K. Y. Wong and F. M. Wahl, "Document analysis system," *IBM Journal of Research and Development*, vol. 26, pp. 647–656, 1982.

[4] J. Fisher, "A rule-based system for document image segmentation," in *Pattern Recognition, 10th international conference*, pp. 567–572, 1990.

[5] L. O'Gorman, "The document spectrum for page layout analysis," vol. 15, no. 11, pp. 1162–1173, 1993.

[6] Y. Chen and B. Wu, "A multi-plane approach for text segmentation of complex document images," *Pattern Recognition*, vol. 42, no. 7, pp. 1419–1444, 2009.

[7] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Pearson Education, 3 ed., 2008.

[8] F. Shafait, D. Keysers, and T. Breuel, "Performance evaluation and benchmarking of six-page segmentation algorithms," vol. 30, no. 6, pp. 941–954, 2008.

[9] K. Jung, K. Kim, and A. K. Jain, "Text information extraction in images and video: a survey," *Pattern Recognition*, vol. 37, no. 5, pp. 977–997, 2004.

[10] G. Nagy, "Twenty years of document image analysis in PAMI," vol. 22, no. 1, pp. 38–62, 2000.

[11] A. K. Jain and B. Yu, "Document representation and its application to page decomposition," vol. 20, no. 3, pp. 294–308, 1998.

[12] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE trans. on Systems, Man and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[13] H. Oh, K. Lim, and S. Chien, "An improved binarization algorithm based on a water flow model for document image with inhomogeneous backgrounds," *Pattern Recognition*, vol. 38, no. 12, pp. 2612–2625, 2005.

[14] W. Niblack, *An introduction to Digital Image Processing*. Prentice-Hall, 1986.

[15] J. Sauvola and M. Pietaksinen, "Adaptive document image binarization," *Pattern Recognition*, vol. 33, no. 2, pp. 335–236, 2000.

[16] J. Kapur, P. Sahoo, and A. Wong, "A new method for gray-level picture thresholding using the entropy of the histogram," *Graphical Models and Image Processing*, vol. 29, no. 3, pp. 273–285, 1985.

[17] W. Tsai, "Moment-preserving thresholding: A new approach," *Graphical Models and Image Processing*, vol. 29, no. 3, pp. 377–393, 1985.

[18] P. Stathis, E. Kavallieratou, and N. Papamarkos, "An evaluation survey of binarization algorithms on historical documents," in *19th International Conference on Pattern Recognition*, pp. 1–4, 2008.

[19] J. M. White and G. D. Rohrer, "Image thresholding for optical character recognition and other applications requiring character image extraction," *IBM J. Res. Dev*, vol. 27, pp. 400–411, 1983.

[20] M. Kamel and A. Zhao, "Binary character/graphics image extraction: a new technique and six evaluation aspects," *IAPR International conference*, vol. 3, no. C, pp. 113–116, 1992.

[21] Y. Liu and S. Srihari, "Document image binarization based on texture features," vol. 19, no. 5, pp. 540–544, 1997.

[22] R. de Queiroz, Z. Fan, and T. Tran, "Optimizing block-thresholding segmentation for multilayer compression of compound images," vol. 9, no. 9, pp. 1461–1471, 2000.

[23] H. Cheng and C. A. Bouman, "Document compression using rate-distortion optimized segmentation," *Journal of Electronic Imaging*, vol. 10, no. 2, pp. 460–474, 2001.

[24] L. Bottou, P. Haffner, P. G. Howard, P. Simard, Y. Bengio, and Y. LeCun, "High quality document image compression with DjVu," *Journal of Electronic Imaging*, vol. 7, no. 3, pp. 410–425, 1998.

[25] P. Haffner, L. Bottou, and Y. Lecun, "A general segmentation scheme for DjVu document compression," in *Proc. of ISMM 2002*, (Sydney, Australia), April 2002.

[26] "Luradocument pdf compressor." Available from https://www.luratech.com.

[27] J. Besag, "On the statistical analysis of dirty pictures," *J. Roy. Statist. Soc. B*, vol. 48, no. 3, pp. 259–302, 1986.

[28] Y. Zheng and D. Doermann, "Machine printed text and handwriting identification in noisy document images," vol. 26, no. 3, pp. 337–353, 2004.

[29] S. Kumar, R. Gupta, N. Khanna, S. Chaundhury, and S. D. Joshi, "Text extraction and document image segmentation using matched wavelets and MRF model," vol. 16, no. 8, pp. 2117–2128, 2007.

[30] J. Kuk, N. Cho, and K. Lee, "MAP-MRF approach for binarization of degraded document image," in *Proc. of IEEE Int'l Conf. on Image Proc.*, pp. 2612–2615, 2008.

[31] H. Cao and V. Govindaraju, "Processing of low-quality handwritten documents using Markov Random Field," vol. 31, no. 7, pp. 1184–1194, 2009.

[32] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. 18th International Conf. on Machine Learning*, pp. 282–289, Morgan Kaufmann, 2001.

[33] S. Kumar and M. Hebert, "Discriminative random fields: A discriminative framework for contextual interaction in classification," in *Proc. of Int'l Conference on Computer Vision*, vol. 2, pp. 1150–1157, 2003.

[34] M. C.-P. a. X. He, R.S. Zemel, "Multiscale conditional random fields for image labeling," *Proc. of IEEE Computer Soc. Conf. on Computer Vision and Pattern Recognition*, vol. 2, pp. 695–702, 2004.

[35] M. Li, M. Bai, C. Wang, and B. Xiao, "Conditional random field for text segmentation from images with complex background," *Pattern Recognition Letters*, vol. 31, no. 14, pp. 2295–2308, 2010.

[36] M. Sezgin and B. Sankur, "Survey over image thresholding techiniques and quantitative performance evaluation," *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 146–165, 2004.

[37] H. Cheng, G. Feng, and C. A. Bouman, "Rate-Distortion based segmentation for MRC compression," in *Proc. of SPIE Conf. on Color Imaging: Device-Independent Color, Color Hardcopy and Applications*, vol. 4663, (San Jose, CA), 21-23 January 2002.

[38] E. Haneda, J. Yi, and C. A. Bouman, "Segmentation for MRC compression," in *Color Imaging XII: Processing, Hardcopy, and Applications*, vol. 6493, (San Jose, CA), 29th January 2007.

[39] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. McGraw-Hill, 2 ed., 2001.

[40] J. Besag, "Spatial interaction and the statistical analysis of lattice systems," *Journal of the Royal Statistical society B*, vol. 36, no. 2, pp. 192–236, 1974.

[41] S. Z. Li, *Markov Random Fields modeling in image analysis*. Springer, second ed., 2001.

[42] C. A. Bouman, "Digital image processing laboratory: Markov random fields and MAP image segmentation." Available from http://www.ece.purdue.edu/~bouman, January 2007.

[43] C. A. Bouman, "Cluster: An unsupervised algorithm for modeling Gaussian mixtures." Available from http://www.ece.purdue.edu/~bouman, April 1997.

[44] J. Rissanen, "A universal prior for integers and estimation by minimum description length," *The annals of Statistics*, vol. 11, no. 2, pp. 417–431, 1983.

[45] J. Besag, "Statistical analysis of non-lattice data," *The Statistician*, vol. 24, no. 3, pp. 179–195, 1975.

[46] J. Besag, "Efficiency of pseudolikelihood estimation for simple Gaussian fields," *Biometrika*, vol. 64, no. 3, pp. 616–618, 1977.

[47] J. Lagarias, J. Reeds, M. Wright, and P. Wright, "Convergence properties of the nelder-mead simplex method in low dimensions," *SLAM Journal of Optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[48] C. A. Bouman and B. Liu, "Multiple Resolution Segmentation of Textured Images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 13, no. 2, pp. 99–113, 1991.

[49] C. A. Bouman and M. Shapiro, "A multiscale random field model for Bayesian image segmentation," *IEEE Trans. on Image Processing*, vol. 3, no. 2, pp. 162–177, 1994.

[50] H. Cheng and C. A. Bouman, "Multiscale Bayesian Segmentation Using a Trainable Context Model," *IEEE Trans. on Image Processing*, vol. 10, no. 4, pp. 511–525, 2001.

[51] B. Gidas, "A renormalization group approach to image processing problems," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 11, no. 2, pp. 164–180, 1989.

[52] M. Comer and E. J. Delp, "Segmentation of Textured Images Using a Multiresolution Gaussian Autoregressive Model," *IEEE Trans. on Image Processing*, vol. 8, no. 3, pp. 408–420, 1999.

[53] H. Cheng and C. A. Bouman, "Multiscale Bayesian segmentation using a trainable context model," *IEEE trans. on image processing*, vol. 10, no. 4, pp. 511–525, 2001.

[54] D. Mukherjee, N. Memon, and A. Said, "JPEG-matched MRC compression of compound documents," in *Proc. of IEEE Int'l Conf. on Image Proc.*, vol. 3, pp. 434–437, 7-10 October 2001.

[55] D. Mukherjee, C. Chrysafis, and A. Said, "JPEG2000-matched MRC compression of compound documents," in *Proc. of IEEE Int'l Conf. on Image Proc.*, vol. 3, pp. 73–76, 24-28 June 2002.

[56] R. de Queiroz, "On data filling algorithms for MRC layers," in *Proc. of IEEE Int'l Conf. on Image Proc.*, vol. 2, pp. 586–589, 10-13 September 2000.

[57] G. Pavlidis, S. Tsekeridou, and C. Chamzas, "JPEG-matched data filling of sparse images," in *Proc. of IEEE Int'l Conf. on Image Proc.*, vol. 1, pp. 493–496, 24-27 October 2004.

[58] L. Bottou, P. Haffner, and P. Howard, "High quality document image compression with DjVu," *Journal of Electronic Imaging*, vol. 7, no. 3, pp. 410–425, 1998.

[59] "Document express with djvu." Available from https://www.celartem.com.

[60] International Electrotechinical Commission, *IEC 61966-2-1.* 1999.

[61] H. Siddiqui and C. A. Bouman, "Training-Based Descreening," *IEEE Trans. on Image Processing*, vol. 16, no. 3, pp. 789–802, 2007.

[62] A. Osman, Z. Pizlo, and J. Allebach, "CRT calibration techniques for better accuracy including low luminance colors," in *Proc. of SPIE Conf. on Color Imaging: Processing, Hardcopy and Applications*, vol. 5293, (San Jose), pp. 286–297, January 2004.

[63] G. Sharma, *Digital Color Imaging*. CRC press, 1 ed., 2002.

[64] Kodak Technical information are available from ftp://ftp.kodak.com/GASTDS/Q60DATA/TECHINFO.pdf, June 2003.

[65] International Telecommunication Union, *ITU-T recommendation T.88 Lossy/lossless coding of bi-level images*. February 2000.

[66] P. G. Howard, F. Kossentini, B. Martins, S. Forchhammer, W. J. Rucklidge, and F. Ono, "The emerging JBIG2 standard."

[67] H. Akaike, "A New Look at the Statistical Model Identification," *IEEE Trans. on Automatic Control*, vol. 19, no. 6, pp. 716–723, 1974.

[68] G. Sherma, "Cancellation of show-through in duplex scanning," in *Proc. of IEEE Int'l Conf. on Image Proc.*, pp. 609–612, 2000.

[69] K. Barnard, V. Carder, and B. Funt, "A comparison of computational color constancy algorithms - part 1: methodology and experiments with synthesized data," *IEEE Trans. on Image Processing*, vol. 11, no. 9, pp. 972–984, 2002.

[70] G. Buchsbaum, "A spatial processor model for object colour perception," *J. Frankilin inst.*, vol. 310, pp. 1–26, 1980.

[71] "Tutorials for computer vision and computational color vision." Available from http://www.cs.sfu.ca/~colour/research/colour-constancy.html.

[72] D. Forsyth, "A novel algorithm for colour constancy," *Int'l J. Computer Vision*, vol. 5, no. 1, pp. 5–36, 1990.

[73] G. Finlayson, S. Hordley, and P. Hubel, "Color by correlation: A simple, unifying framework for color constancy," vol. 23, pp. 1209–1221, 2001.

[74] M. D. Fairchild, *Color Appearance Models*. Addison-Wesley, 1998.

[75] H. R. Kang, "Computational color technology," in *SPIE press*, 2006.

[76] Anil K. Jain, *Fundamentals of digital image processing*. Prentice hall, 1988.

APPENDIX

A. FEATURE VECTOR FOR CCC

The feature vector for the connected components (CC) extracted in the CCC algorithm is a 4-dimensional vector and denoted as $y = [y_1\ y_2\ y_3\ y_4]^T$. Two of the components describe edge depth information, while the other two describe pixel value uniformity.

More specifically, an inner pixel and an outer pixel are first defined to be the two neighboring pixels across the boundary for each boundary segment $k \in \{1, \ldots, N\}$, where N is the length of the boundary. Note that the inner pixel is a foreground pixel and the outer pixel is a background pixel. The inner pixel values are defined as $X_{in}(k) = [R_{in}(k),\ G_{in}(k),\ B_{in}(k)]$, whereas the outer pixel values are defined as $X_{out}(k) = [R_{out}(k),\ G_{out}(k),\ B_{out}(k)]$. Using these definitions, the edge depth is defined as

$$edge(k) = \sqrt{||X_{in}(k) - X_{out}(k)||^2}.$$

Then, the terms y_1 and y_2 are defined as,

$$y_1 \stackrel{\text{def}}{=} \text{Sample mean of } edge(k),\ k = 1, 2, \ldots, N$$

$$y_2 \stackrel{\text{def}}{=} \text{Standard deviation of } edge(k),\ k = 1, 2, \ldots, N.$$

The terms y_3 and y_4 describe uniformity of the outer pixels. The uniformity is defined by the range and standard deviation of the outer pixel values, that is

$$y_3 \stackrel{\text{def}}{=} \max\{O(k)\} - \min\{O(k)\},\ k = 1, 2, \ldots, N$$

$$y_4 \stackrel{\text{def}}{=} \text{Standard deviation of } O(k),\ k = 1, 2, \ldots, N$$

where

$$O(k) = \sqrt{R_{out}(k)^2 + G_{out}(k)^2 + B_{out}(k)^2}.$$

In an actual calculation, the 95^{th} percentile and the 5^{th} percentile are used instead of the maximum and minimum values to eliminate outliers. Note that only outer pixel

values were examined for the uniformness because we found that inner pixel values of the connected components extracted by COS are mostly uniform even for non-text components.

The augmented feature vector of CCC algorithm contains the four components described above concatenated with two additional components corresponding the horizontal and vertical position of the connected component's center in 300 dpi, that is $z = [y_1\ y_2\ y_3\ y_4\ a_1\ a_2]^T$.

$a_1 \stackrel{\text{def}}{=}$ horizontal pixel location of a connected component's center

$a_2 \stackrel{\text{def}}{=}$ vertical pixel location of a connected component's center