DOCUMENT IMAGE SEGMENTATION AND COMPRESSION

A Thesis

Submitted to the Faculty

of

Purdue University

by

Hui Cheng

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 1999

To my beloved wife Liu, Qian.

To my wonderful parents Cheng, Zuoqin and Li, Heying.

ACKNOWLEDGMENTS

I would like to extend my most sincere thanks to my advisor, Professor Charles A. Bouman for his guidance, encouragement and all the things that he had done in helping me develop my professional and personal skills. I am certain that I will benefit from his rigorous scientific approach, and the way of critical thinking throughout my future career.

Most of all, my deepest thanks go to my wife Qian, my parents and my family. I can not thank them enough for their love, support, sacrifice and their belief in me.

I want to thank my advisory committee members: Professor Jan P. Allebach, Professor Edward J. Delp, and Professor Bradley J. Lucier for their constructive suggestions and comments. Also, my thanks go to Dr. Zhigang Fan, Dr. Ricardo L. de Queiroz, Dr. Chi-hsin Wu and Dr. Steve J. Harrington of Xerox Corporation for their valuable advice and suggestions. I thank Dr. Faouzi Kossentini and Mr. Dave Tompkins of Department of Electrical and Computer Engineering, University of British Columbia for providing us the JBIG2 coder. In addition, I am grateful to all my friends who gave me help, support, and encouragement. Thank you all!

I would also like to thank Xerox Corporation, Xerox Foundation, and Xerox IMPACT Imaging for their generous financial support. I thank ASEE, ASEE Prism, IEEE, IEEE Spectrum, and Stanley Electric Sales of America for allowing me to use their documents published on ASEE Prism and IEEE Spectrum in this research.

TABLE OF CONTENTS

Page

LIST OF TABLES vii

LIST OF FIGURES ix

ABSTRACT xi

1 Introduction 1

2 Trainable Sequential MAP Segmentation Algorithm 5

2.1 Introduction 5

2.2 Multiscale Image Segmentation 9

2.3 Computing the SMAP Estimate 12

2.3.1 Computing Context Terms for the SMAP Estimate 13

2.3.2 Computing Log Likelihood Terms for SMAP Estimate 15

2.4 Parameter Estimation 18

2.4.1 Estimation of Context Model Parameters 19

2.4.2 Estimation of Quadtree Parameters 22

2.4.3 Decimation of Ground Truth Segmentation 23

2.4.4 Estimation of Data Model Parameters 24

2.5 Experimental Results 24

2.6 Conclusion 26

3 Document Compression Using Rate-Distortion Optimized Segmentation . . 35

3.1 Introduction 35

3.2 Multilayer Compression Algorithm 39

3.2.1 Compression of One-color Blocks 41

3.2.2 Compression of Two-color Blocks 41

3.2.3 Compression of Picture Blocks and Other Blocks 43

3.2.4 Additional Issues 44

3.2.5 Use of the TSMAP Segmentation Algorithm 45
3.3 Rate-Distortion Optimized Segmentation 46
3.3.1 Estimate Bit Rates and Distortion of One-color Blocks 48
3.3.2 Estimate Bit Rates and Distortion of Two-color Blocks 48
3.3.3 Estimate Bit Rates and Distortion of JPEG Blocks 51
3.4 Experimental Results 53
3.5 Conclusion 57
LIST OF REFERENCES 67
APPENDICES 73
Appendix A: Computing Log Likelihood Terms 73
Appendix B: Computation of EM Update Using Stochastic Sampling . . . 73
VITA 75

LIST OF TABLES

Table Page

3.1 Bit rates, compression ratios and RDOS distortion of images compressed using both TSMAP and RDOS 54

3.2 Average bit rate of coding each class 55

LIST OF FIGURES

Figure		Page
2.1	Bayesian segmentation approach	9
2.2	Multiscale Bayesian segmentation approach	9
2.3	Pyramidal graph model	13
2.4	Class probability tree	14
2.5	1-D analog of the quadtree model	16
2.6	Parameter estimation of the context model	19
2.7	Splitting rule based on least squares estimation	20
2.8	Dependency among class labels in the quadtree model	23
2.9	Decimation of the ground truth	28
2.10	Training images and their ground truth segmentations	29
2.11	Comparison of segmentation results among different algorithms	30
2.12	TSMAP segmentation results I	31
2.13	TSMAP segmentation results II	32
2.14	Effect of the number of training images on TSMAP	33
3.1	General structure of the multilayer document compression algorithm	39
3.2	Flow diagram of the multilayer document compression algorithm	40
3.3	Minimal MSE thresholding	42
3.4	Two-color distortion measure	50
3.5	Segmentation results of TSMAP and RDOS	59
3.6	Comparison between images compressed using TSMAP and RDOS at similar bit rates.	60
3.7	RDOS segmentations with different λ's	60
3.8	Comparison of rate-distortion performance of the multilayer compression algorithm using RDOS, TSMAP and manual segmentations	61

3.9 Test image III and its segmentations 61
3.10 Compression result I 62
3.11 Compression result II 63
3.12 Compression result III 64
3.13 Compression result IV 65
3.14 Estimated vs. true bit rates of coding each class 66

ABSTRACT

Cheng, Hui, Ph.D., Purdue University, August, 1999. Document Image Segmentation and Compression. Major Professor: Charles A. Bouman.

In the first part of this research, we propose an image segmentation algorithm called the trainable sequential MAP (TSMAP) algorithm. The TSMAP algorithm is based on a multiscale Bayesian approach. It has a novel multiscale context model which can capture complex aspects of both local and global contextual behavior. In addition, its image model uses local texture features extracted via a wavelet decomposition, and the textural information at various scales is captured by a hidden Markov model. The parameters which describe the characteristics of typical images are extracted from a database of training images and their accurate segmentations. Once the training procedure is performed, scanned documents may be segmented using a fine-to-coarse-to-fine procedure that is computationally efficient.

In the second part of this research, we introduce a multilayer compression algorithm for document images. This compression algorithm first segments a scanned document image into different classes, then compresses each class using an algorithm specifically designed for that class. We also propose a rate-distortion optimized segmentation (RDOS) algorithm developed for document compression. Compared with the TSMAP algorithm, the RDOS algorithm can often result in a better rate-distortion trade-off, and produce more robust segmentations than TSMAP by eliminating those misclassifications which can cause severe artifacts. Experimental results show that, at similar bit rates, the multilayer compression algorithm using RDOS can achieve a much higher subjective quality than well-known coders such as DjVu, SPIHT, and JPEG.

1. Introduction

With the advent of modern publishing technologies, the layout of today's documents has never been more complex. Most of them contain not only text and background regions, but also graphics, tables and pictures. Therefore scanned documents must often be segmented before other document processing techniques, such as compression or rendering, can be applied.

Traditional approaches to document segmentation, usually involve partitioning the document images into blocks, and then classifying each block [1, 2, 3]. Early works of the block-based approaches are mainly designed for binary document images. For example, Wong, Casey and Wahl [1] proposed a technique called the run length smoothing algorithm (RLSA) to partition a binary document image into blocks. Each block was then classified as text or picture according to some statistical features, such as the horizontal white-black transitions of the image data. A similar algorithm was also investigated by Wang et al. for newspaper layout analysis [2]. Chauvet and coworkers [3] presented a recursive block partition algorithm based on RLSA. They used the linear closing with variable length structuring elements to extract features for block classification. A more detailed survey of these approaches can be found in [4].

Recent block-based segmentation algorithms are developed mostly for grayscale or color document images. Among these algorithms, some use features extracted from the discrete cosine transform (DCT) coefficients to separate text blocks from picture blocks. For example, Murata [5] proposed a method based on the absolute values of DCT coefficients, and Konstantinides and Tretter [6] use a DCT block activity measure. Other block-based segmentation algorithms extract features directly from the document image. In [7], text and line graphics are extracted from check images

using morphological filters followed by thresholding. Ramos and de Queiroz proposed a block-based activity measure as a feature for separating edge blocks, smooth blocks and detailed blocks for document coding [8].

Alternatively, texture based approaches [9, 10, 11] treat different components of a document image as different textures. The scanned document images are first convolved with a set of masks to generate feature vectors. Each feature vector is then classified into different classes using a pre-trained classifier, such as a neural network [9, 11].

In Chapter 2, we propose a new algorithm for document segmentation which is call the Trainable Sequential MAP (TSMAP) segmentation algorithm. The TSMAP algorithm is a general purpose image segmentation algorithm, and it is based on the multiscale Bayesian framework proposed by Bouman and Shapiro [12]. TSMAP exploits both local texture characteristics and image structure to segment the scanned documents into different regions such as text, background, and pictures. It has a novel multiscale context model which can capture complex aspects of both local and global contextual behavior. The method is based on the use of tree classifiers [13] to model the transition probabilities between adjacent scales in the multiscale structure. In addition, TSMAP has a multiscale image model which uses local texture features extracted via a wavelet decomposition. The textural information at various scales is then captured through a hidden Markov model, and the dependence of features between adjacent scales is extracted using inter-scale prediction.

The parameters needed for both the image model and the context model are estimated from a database of training images which are produced by scanning typical documents and manually segmenting them into desired components. Once the training procedure is performed, scanned documents may be segmented using a fine-to-coarse-to-fine procedure that is computationally efficient.

In Chapter 3, we will discuss document image compression, and rate-distortion optimized segmentation for document compression. During the last decade, high quality document images have been used in many image processing systems, such as digital

color copiers, color FAX machines and digital libraries, where paper documents are digitally scanned, stored, transmitted and then printed or displayed. Typically, these operations must be performed rapidly, and user expectations of quality are very high since the final output is often subject to close inspection. Digital implementation of this imaging pipeline is particularly formidable when one considers that a single page of a color document scanned at 400-600 dpi (dots per inch) requires approximately 45-100 Megabytes of storage. Consequently, practical systems for processing color documents require document compression methods that achieve high compression ratios and with very low distortion.

A unique property of document images is that they consist of regions with distinct characteristics, such as text, picture and background. Typically, text requires high spatial resolution for legibility, but does not require high color resolution. On the other hand, continuous-tone pictures need high color resolution, but can tolerate low spatial resolution. Therefore, a good document compression algorithm must be spatially adaptive, in order to meet different needs and exploit different types of redundancy among different image classes. Traditional compression algorithms, such as JPEG, are based on the assumption that the input image is spatially homogeneous, so they tend to perform poorly on document images.

In Chapter 3, we introduce a multilayer compression algorithm for document images. This algorithm first classifies 8×8 non-overlapping blocks of pixels into different classes. Then, each class is compressed using an algorithm specifically designed for that class. We also propose a rate-distortion optimized segmentation (RDOS) algorithm designed to work with document compression. The RDOS algorithm works in a closed loop fashion by applying each coding method to each region of the document and then selecting the method that yields the best rate-distortion trade-off. The RDOS optimization is based on the measured distortion and an estimate of the bit rate for each coding method. Compared with the TSMAP algorithm, the RDOS algorithm can often result in a better rate-distortion trade-off, and produce more robust segmentations than TSMAP by eliminating those misclassifications which can cause

severe artifacts. Experimental results show that, at similar bit rates, the multilayer compression algorithm using RDOS can achieve a much higher subjective quality than well-known coders such as DjVu, SPIHT, and JPEG.

2. Trainable Sequential MAP Segmentation Algorithm

In recent years, multiscale Bayesian approaches have attracted increasing attention for use in image segmentation. Generally, these methods tend to offer improved segmentation accuracy with reduced computational burden. Existing Bayesian segmentation methods use simple models of context designed to encourage large uniformly classified regions. Consequently, these context models have a limited ability to capture the complex contextual dependencies that are important in applications such as document segmentation.

In this chapter, we propose a multiscale Bayesian segmentation algorithm which can effectively model complex aspects of both local and global contextual behavior. The model uses a Markov chain in scale to model the class labels that form the segmentation, but augments this Markov chain structure by incorporating tree based classifiers to model the transition probabilities between adjacent scales. The tree based classifier models complex transition rules with only a moderate number of parameters.

One advantage to our segmentation algorithm is that it can be trained for specific segmentation applications by simply providing examples of images with their corresponding accurate segmentations. This makes the method flexible by allowing both the context and the image models to be adapted without modification of the basic algorithm. We illustrate the value of our approach with examples from document segmentation in which text, picture and background classes must be separated.

2.1 Introduction

Image segmentation is an important first step for many image processing applications. For example, in document processing it is usually necessary to segment out text, picture and graphic regions before scanned documents can be effectively analyzed, compressed or rendered [1, 4]. Segmentation has also been shown useful for

image and video compression [14, 15]. For each of these cases, the objective is to separate images into regions with distinct homogeneous behavior.

In recent years, Bayesian approaches to segmentation have become popular because they form a natural framework for integrating both statistical models of image behavior and prior knowledge about the contextual structure of accurate segmentations. An accurate model of contextual structure can be very important for segmentation. For example, it may be known that segmented regions must have smooth boundaries or that certain classes can not be adjacent to one another.

In a Bayesian framework, contextual structure is often modeled by a Markov random field (MRF) [16, 17, 18]. Usually, the MRF contains the discrete class of each pixel in the image. The objective then becomes to estimate the unknown MRF from the available data. In practice, the MRF model typically encourages the formation of large uniformly classified regions. Generally, this smoothing of the segmentation increases segmentation accuracy, but it can also smear important details of a segmentation, and distort segmentation boundaries. Approaches based on MRF's also tend to suffer from high computational complexity. The non-causal dependence structure of MRF's usually results in iterative segmentation algorithms, and can make parameter estimation difficult [19, 20]. Moreover, since the true segmentation is not available, parameter estimation must be done using an incomplete data method such as the EM algorithm [21, 22, 23].

Another long term trend has been the incorporation of multiscale techniques in segmentation algorithms. Methods such as pyramid pixel linking [24], boundary refinement [25, 26], and decision integration [27] have been used to enforce contextual information in the segmentation process. In addition, pyramid [28] or wavelet decompositions [29, 30] yield powerful multiscale features that can capture both local and global image characteristics.

Not surprisingly, there has been considerable interest in combining both Bayesian and multiscale techniques into a single framework. Initial attempts to merge these view points focused on using multiscale algorithms to compute segmentations but

retained the underlying fixed scale MRF context model [31, 32, 33]. These researchers found that multiscale algorithms could substantially reduce computation and improve robustness, but the simple MRF context model limited the quality of segmentations.

In [34, 12], Bouman and Shapiro introduced a multiscale context model in which the segmentation was modeled using a Markov chain in scale. By using a Markov chain, this approach avoided many of the difficulties associated with noncausal MRF structures and resulted in a non-iterative segmentation algorithm similar in concept to the forward-backward algorithm used with hidden Markov models (HMM). Laferte, Heitz, Perez and Fabre used a similar approach, but incorporated a multiscale feature model using a pyramid image decomposition [35]. In related work, Crouse, Nowak, and Baraniuk have proposed the use of multiscale HMM's to model wavelet coefficients for applications such as image de-noising and signal detection [36].

In another approach, Kato, Berthod, and Zerubia first used a 3-D MRF as a context model for segmentation [37]. In this model, each class label depends on class labels at both the same scale and the adjacent finer and coarser scales. Comer and Delp used a similar context model but incorporated a 3-D autoregressive feature model [38].

In this chapter, we propose an image segmentation method based on the multiscale Bayesian framework. Our approach uses multiscale models for both the data and the context. Once a complete model is formulated, the sequential maximum *a posterior* (SMAP) estimator [12] is used to segment images.

An important contribution of our approach is that we introduce a multiscale context model which can capture complex aspects of both local and global contextual behavior. The method is based on the use of tree based classifiers [13, 39] to model the transition probabilities between adjacent scales in the multiscale structure. This multiscale structure is similar to previously proposed segmentation models [12, 40, 41], with the segmentations at each resolution forming a Markov chain in scale. However, the tree based classifier allows for much more complex transition rules, with only a moderate number of parameters. Moreover, we propose an efficient parameter

estimation algorithm for training which is not iterative and needs only one coarse-to-fine recursion through resolutions.

Our multiscale image model uses local texture features extracted via a wavelet decomposition. The wavelet transform produces a pyramid of feature vectors with each three dimensional feature vector representing the texture at a specific location and scale. While wavelet decompositions tend to decorrelate data, significant correlation can remain among wavelet coefficients at similar locations but different scales. In fact, this dependency is often exploited in image coding techniques such as zerotrees [42]. We account for these dependencies by modeling the wavelet feature vectors as a class dependent multiscale autoregressive process [43]. This approach more accurately models some textures without adding significant additional computation.

A feature of our segmentation method is that it can be trained for any segmentation application by simply providing examples of images with their corresponding accurate segmentations. We believe that this makes the method flexible by allowing it to be adapted for different segmentation applications without modification of the basic algorithm. The training procedure uses the example images together with their segmentations to estimate all parameters of both the image and context models in a fully automatic manner (Software implementation of this algorithm is available from http://www.ece.purdue.edu/∼bouman.). Once the model parameters are estimated, segmentation is computationally efficient requiring a single fine-to-coarse-to-fine iteration through the pyramid.

In order to test the performance of our algorithm, we apply it to the problem of document segmentation. This application is interesting because of both its practical significance and the great contextual complexity inherent to modern documents [4]. For example, most documents conform to complex rules regarding the spatial placement of regions such as picture, text, graphics and background. While specifying these rules explicitly would be difficult and error prone, we show that these rules can be effectively learned from a limited number of training examples.

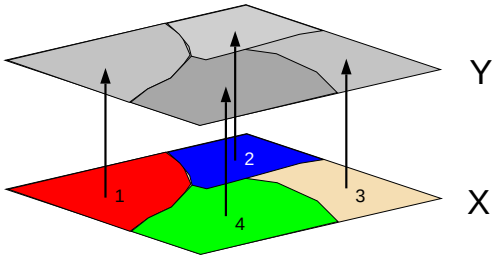


Fig. 2.1. This figure illustrates the approach to Bayesian segmentation. Y is an observed image and X is a random field which contains the class of each pixel in Y. The objective is then to estimate X from Y.

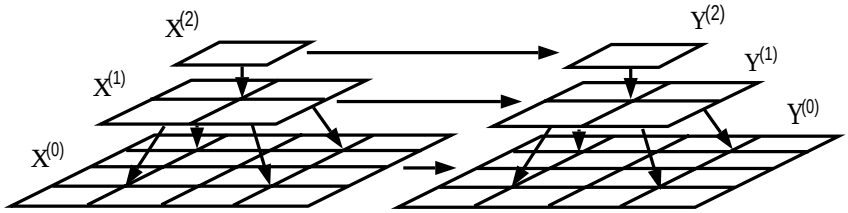


Fig. 2.2. The multiscale segmentation model. $Y^{(n)}$ contains the image feature vectors extracted at scale n while $X^{(n)}$ contains the corresponding class of each pixel at scale n. Notice that both image features, Y, and the context model, X, use multiscale pyramid structures.

2.2 Multiscale Image Segmentation

Figure 2.1 illustrates the basic approach to Bayesian segmentation. The image or its extracted features are denoted by Y, and X represents the discrete random field containing the class of each pixel. The data model is then embodied in the probability density $p_{y|x}(y|x)$, while the prior density $p_x(x)$ is used to incorporate knowledge about the contextual structure of accurate segmentations. In the Bayesian approach, the correct segmentation is then estimated by using the posterior distribution $p_{x|y}(x|y)$.

In this chapter, we will adopt a Bayesian approach, but our method differs from many in that we use a multiscale model for both the data and the context. Figure 2.2 illustrates the basic structure of our multiscale segmentation model [41]. At each scale n, there is a random field of image feature vectors, $Y^{(n)}$, and a random field of

class labels, $X^{(n)}$.[1] For our application, the image features $Y^{(n)}$ will correspond to Haar basis wavelet coefficients at scale n. Intuitively, $Y^{(n)}$ contains image texture and edge information at scale n, while $X^{(n)}$ contains the corresponding class labels. The behavior of $Y^{(n)}$ is therefore assumed dependent on its class labels $X^{(n)}$ and coarse scale image features $Y^{(n+1)}$ as is indicated by the arrows in Figure 2.2.

Notice that each random field $X^{(n)}$ depends on the previous coarser scale field $X^{(n+1)}$. This dependence gives $X^{(n)}$ a Markov chain structure in the scale variable n. We will see that this structure is desirable because it can capture complex spatial dependencies in the segmentation, but it allows for efficient computational processing. The multiscale structure can also account for both large and small scale characteristics that may be desirable in a good segmentation.

For the convenience, we define $X^{(\leq n)} = \{X^{(i)}\}_{i=0}^{n}$ to be the set of class labels at scales n or finer, and $X^{(>n)} = \{X^{(i)}\}_{i=n+1}^{L}$ where L is the coarsest scale. We also define $Y^{(\leq n)}$ and $Y^{(>n)}$ similarly. Using this notation, the Markov chain structure may be formally expressed in terms of the probability mass functions

$$p_{x^{(n)}|x^{(>n)}}(x^{(n)}|x^{(>n)}) = p_{x^{(n)}|x^{(n+1)}}(x^{(n)}|x^{(n+1)}) \ . \tag{2.1}$$

So the probability of x is given by

$$p_x(x) = \prod_{n=0}^{L} p_{x^{(n)}|x^{(n+1)}}(x^{(n)}|x^{(n+1)}) \tag{2.2}$$

where throughout this chapter the term $p_{x^{(L)}|x^{(L+1)}}(x^{(L)}|x^{(L+1)})$ is assumed to mean $p_{x^{(L)}}(x^{(L)})$ since L is the coarsest scale.

The image features $y^{(n)}$ are assumed conditionally independent given the class labels $x^{(n)}$ and image features $y^{(n+1)}$ at the coarser scale. Therefore, the conditional density of y given x may be expressed as

$$p_{y|x}(y|x) = \prod_{n=0}^{L} p_{y^{(n)}|x^{(n)},y^{(n+1)}}(y^{(n)}|x^{(n)}, y^{(n+1)}) \tag{2.3}$$

[1]We will use upper case letters to denote random quantities while lower case variables will denote their realizations.

Combining equations (2.2) and (2.3) results in the joint density

$$\begin{aligned} p_{y,x}(y,x) &= p_{y|x}(y|x)\, p_x(x) \\ &= \prod_{n=0}^{L} p_{y^{(n)}|x^{(n)},y^{(n+1)}}(y^{(n)}|x^{(n)},y^{(n+1)}) p_{x^{(n)}|x^{(n+1)}}(x^{(n)}|x^{(n+1)}) \ . \end{aligned}$$

In order to segment the image, we must estimate the class labels X from the image feature data Y. Perhaps the MAP estimator is the most common method for doing this. However, the MAP estimate is not well behaved for multiscale segmentation because it results from minimization of a cost functional which equally weights both fine and coarse scale misclassifications. In practice, coarse scale misclassifications are much more important since they affect many more pixels.

We will therefore use the sequential MAP (SMAP) estimator proposed in [12]. Formally, the SMAP segmentation, $\hat{x}^{(n)}$, is computed using the recursive coarse-to-fine relationship

$$\hat{x}^{(n)} = \arg\max_{x^{(n)}} \left\{ \log p_{y^{(\leq n)}|x^{(n)},y^{(n+1)}}(y^{(\leq n)}|x^{(n)},y^{(n+1)}) + \log p_{x^{(n)}|\hat{x}^{(n+1)}}(x^{(n)}|\hat{x}^{(n+1)}) \right\} \tag{2.4}$$

where the coarsest segmentation $\hat{x}^{(L)}$ is computed using the conventional MAP estimate. The SMAP estimation procedure is a coarse-to-fine recursion which starts by computing $\hat{x}^{(L)}$, the MAP estimate at the coarsest scale L. At each scale n, equation (2.4) is then applied to compute the new segmentation while conditioning on the previous coarser scale segmentation $\hat{x}^{(n-1)}$. Each application of (2.4) is similar to MAP estimate since it requires maximization of a data term related to $y^{(\leq n)}$ and a context or prior term related to the probability of $x^{(n)}$ conditioned on the previous coarser segmentation $\hat{x}^{(n+1)}$.

In [12], it was shown that the SMAP estimator results from the minimization

$$\hat{x} = \arg\min_{x} E[C(X,x)|Y=y] \tag{2.5}$$

where $C(X,x)$ is the cost of choosing segmentation x when the true segmentation is X, and $C(X,x)$ is chosen to be

$$C(X,x) = \frac{1}{2} + \sum_{n=0}^{L} 2^{n-1} C_n(X,x)$$

$$C_n(X,x) = 1 - \prod_{i=n}^{L} \delta(X^{(i)} - x^{(i)})$$

where $\delta(X^{(i)} - x^{(i)}) = 1$, if $X^{(i)} = x^{(i)}$ and $\delta(X^{(i)} - x^{(i)}) = 0$, if $X^{(i)} \neq x^{(i)}$. While [12] did not assume the same multiscale data model as is used in this chapter, the methods of the proof go through without change. Intuitively, this SMAP cost functional assigns more weight to misclassifications at coarser scales, and is therefore more appropriate for application in discrete multiscale estimation problems.

2.3 Computing the SMAP Estimate

In the previous section, we described a general approach to segmentation. In this section, we will give specific forms for both the data and the context terms of our model, and use these forms to derive a specific algorithm for the SMAP estimator.

Our model will have two important properties. First, we will assume that the data term of (2.4) can be expressed as the sum of log likelihood functions at each pixel. We denote individual pixels by $x_s^{(n)}$ and $y_s^{(n)}$, where s is the position in a 2-D lattice $S^{(n)}$. Using this notation, the data term of (2.4) will have the form

$$\log p_{y^{(\leq n)}|x^{(n)},y^{(n+1)}}(y^{(\leq n)}|x^{(n)},y^{(n+1)}) = \sum_{s\in S^{(n)}} l_s^{(n)}(x_s^{(n)}) \tag{2.6}$$

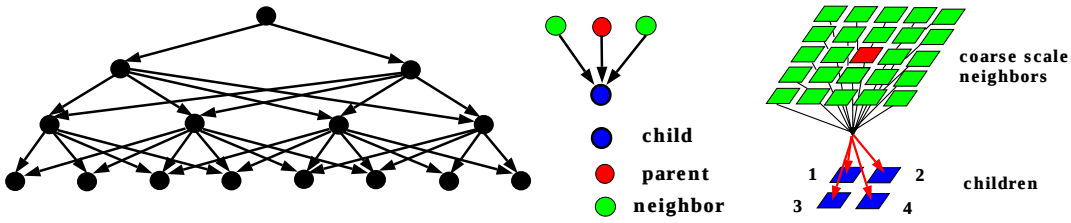
where the functions $l_s^{(n)}(k)$ are appropriately chosen log likelihood functions. Section 2.3.2 will give the details for how to compute these functions $l_s^{(n)}(k)$.

Second, we will assume that the context term of (2.4) can be expressed as the product of probabilities for each pixel. That is the class labels $x_s^{(n)}$ are assumed conditionally independent given the coarser segmentation $x^{(n+1)}$. Therefore, the context term of (2.4) will have the form

$$\log p_{x^{(n)}|x^{(n+1)}}(x^{(n)}|\hat{x}^{(n+1)}) = \sum_{s\in S^(n)} \log\, p_{x_s^{(n)}|x^{(n+1)}}(x_s^{(n)}|\hat{x}^{(n+1)}) \tag{2.7}$$

Section 2.3.1 will give the details for how to compute the conditional probabilities $p_{x_s^{(n)}|x^{(n+1)}}(k|\hat{x}^{(n+1)})$.

With these two assumptions, the SMAP recursion of (2.4) can be simplified to a



(a) (b)

Fig. 2.3. The pyramidal graph model. (a) 1-D analog of the pyramidal graph model, where each pixel has 3 neighbors at the coarser scale. (b) 2-D pyramidal graph model using a 5×5 neighborhood. This is equivalent to interpolation of a pixel at the previous coarser scale into four pixels at the current scale.

single pass, pixel by pixel update rule

$$\hat{x}_s^{(n)} = \arg \max_{0 \le k < M} \left\{ l_s^{(n)}(k) + \log p_{x_s^{(n)}|x^{(n+1)}}(k|\hat{x}^{(n+1)}) \right\} \tag{2.8}$$

where M is the number of possible class labels.

2.3.1 Computing Context Terms for the SMAP Estimate

Our context model requires that we compute the probability distribution for each pixel $x_s^{(n)}$ given the coarser scale segmentation $x^{(n+1)}$. In order to limit complexity of the model, we will assume that $x_s^{(n)}$ is only dependent on $x_{\partial s}^{(n+1)}$, a set of neighboring pixels at the coarser scale. Here, $\partial s \subset S^{(n+1)}$ denotes a window of pixels at scale $n+1$. We will refer this dependency among class labels as the pyramidal graph model. Figure 2.3(a) illustrates the pyramidal graph model for the 1-D case where each pixel has 3 neighbors at the coarser scale. Notice that each arrow points from a neighbor in $x_{\partial s}^{(n+1)}$ to a pixel $x_s^{(n)}$.

Intuitively, this context model is also a model for interpolating a pixel $s^{(n+1)}$ into its child pixels. Figure 2.3(b) illustrates this situation in 2-D when a 5×5 neighborhood is used at the coarser scale. Notice that in 2-D, each pixel $s^{(n+1)}$ has four child pixels at the next finer resolution. Each of the four child pixels will have the same set of neighbors; however they must be modeled using different distributions, because of their different relative positioning. We denote each of these four distinct probability distributions by $p_i^{(n)}(x_s^{(n)}|x_{\partial s}^{(n+1)})$ for $i = 1, 2, 3, 4$. For simplicity, we will

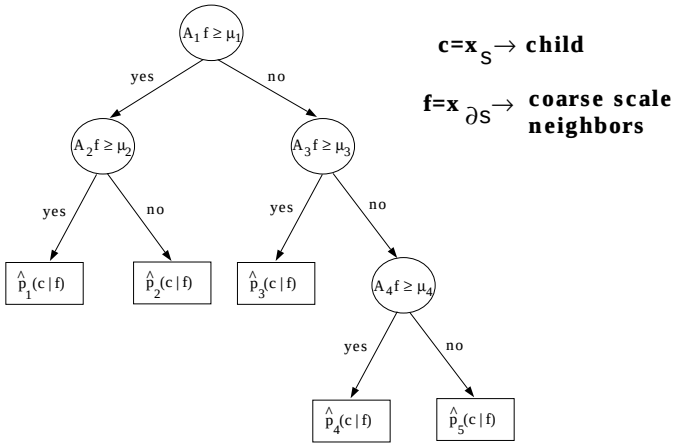


Fig. 2.4. Class probability tree. Circles represent interior nodes, and squares represent leaf nodes. At each interior node, a linear test is performed and the node is split into two child nodes. At each leaf node $\tilde{t}$, the conditional probability mass function $p_i^{(n)}(c|f)$ is approximated by $\hat{p}_{\tilde{t}}(c)$.

use c to denote $x_s^{(n)}$, and f to denote $x_{\partial s}^{(n+1)}$, so that this probability distribution may be written as $p_i^{(n)}(c|f)$. Later we will see that c and f are actually binary encodings of the information contained in $x_s^{(n)}$ and $x_{\partial s}^{(n+1)}$.

Unfortunately, the transition function $p_i^{(n)}(c|f)$ may be very difficult to estimate if the coarse scale neighborhood is large. For example, if there are four classes and the size of the coarse neighborhood is 5×5, there are $4^{25} \approx 10^{16}$ possible values of f. Hence, it is impractical to compute $p_i^{(n)}(c|f)$ using a look-up table containing all possible values of f. For most applications, the distribution of f will be concentrated among a small number of possible values. We can exploit this structure in the distribution of f to dramatically simplify the computation of $p_i^{(n)}(c|f)$.

In order to compute and estimate $p_i^{(n)}(c|f)$ efficiently, we use class probability trees (CPT) [13] to represent $p_i^{(n)}(c|f)$. A CPT is shown in Figure 2.4. The CPT represents a sequence of decisions or tests that must be made in order to compute the conditional probability of c given f. The input to the tree is f. At each interior node, a splitting rule is used to determine which of the two child nodes should be taken. In our case, the splitting rule is computed by comparing $A_t f - \mu_t$ to 0, where A_t is a

pre-computed vector and μ_t is a pre-computed scalar. In this way, f goes down the tree until it reaches a leaf node. Each leaf node $\tilde{t}$ is associated with an empirically computed probability mass function $\hat{p}_{\tilde{t}}(c)$. When f reaches $\tilde{t}$, $p_i^{(n)}(c|f)$ is set to $\hat{p}_{\tilde{t}}(c)$.

If a CPT has K leaf nodes, then the CPT approximates the true transition probability using K probability mass functions. Therefore, by controlling the number of leaf nodes in a CPT, even for a relative large neighborhood, such as a 7×7 neighborhood, we can still estimate the transition probabilities efficiently and accurately. Since a larger neighborhood usually gives more contextual information, CPT's allow us to work with a larger neighborhood and consequently have a better model of the context, while retaining computational efficiency in our model. In section 2.4.1, we will give specific methods for building a CPT from training data.

To achieve the best accuracy from the CPT algorithm, we have found that proper encoding of the quantities $x_s^{(n)}$ and $x_{\partial s}^{(n+1)}$ into c and f is important. Specifically, the encoding should not impose any ordering on the M class labels since this tends to bias the results and consequently to degrade the classification accuracy. We define c to be a binary vector of length M where the $x_s^{(n)}$-th component of c is 1, and other components are 0. If we denote the i-th component of c as c_j, then

$$c_j = \begin{cases} 1 & \text{if } x_s^{(n)} = j \\ 0 & \text{otherwise} \end{cases} \quad 0 \le j < M \ .$$

For example, when $x_s^{(n)} = 2$, and $M = 4$, then $c = (0, 0, 1, 0)$. Similarly, we define f to be a binary vector of length Mb, where b is the number of pixels in the coarse neighborhood ∂s. The binary vector f is then formed by concatenating the binary encodings of each coarse scale neighbor contained in $x_{\partial s}^{(n+1)}$.

2.3.2 Computing Log Likelihood Terms for SMAP Estimate

In order to capture the correlation among image features across scales, we assume that each feature $y_s^{(n)}$ depends on both an image feature $y_{\partial s}^{(n+1)}$ at the coarser scale and its class label $x_s^{(n)}$, where ∂s is the parent of s. We assume that, for each class $x_s^{(n)}$, $y_s^{(n)}$ can be predicted by a different linear function of $y_{\partial s}^{(n+1)}$ which depends on

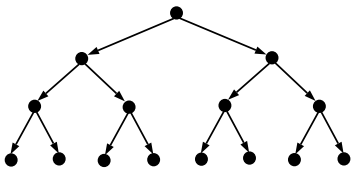


Fig. 2.5. 1-D analog of the quadtree model.

both the class label and the scale. We denote the prediction error by $\tilde{y}_s^{(n)}$.

$$\tilde{y}_s^{(n)} = y_s^{(n)} - \left[\alpha_{x_s}^{(n)} y_{\partial s}^{(n+1)} + \beta_{x_s}^{(n)}\right] \tag{2.9}$$

where $\alpha_{x_s}^{(n)}$ and $\beta_{x_s}^{(n)}$ are prediction coefficients which are functions of both class labels and scales.

To have an efficient algorithm for computing the log likelihood terms $l_s^{(n)}(k)$ defined in equation (2.6), we assume that the prediction errors $\tilde{y}_s^{(n)}$ are conditionally independent given the class labels $x_s^{(n)}$. That is

$$\begin{aligned} \log p_{y^{(n)}|x^{(n)},y^{(n+1)}}(y^{(n)}|x^{(n)},y^{(n+1)}) &= \log p_{\tilde{y}^{(n)}|x^{(n)}}(\tilde{y}^{(n)}|x^{(n)}) \\ &= \sum_{s\in S^{(n)}} \log p_{\tilde{y}_s^{(n)}|x_s^{(n)}}(\tilde{y}_s^{(n)}|x_s^{(n)}) \ . \end{aligned}$$

To calculate the log likelihood terms, we also need to compute the conditional probability distribution of $x_s^{(n)}$ given $x^{(n+1)}$. But we can not use the pyramidal graph model discussed in section 2.3.1, because it will result in a form which is not computationally tractable. Therefore, we use a context model which is simpler than the pyramidal graph model. In this model, we assume that $x_s^{(n)}$ depends only on one class label at the previous coarser resolution. Though we still use $x_{\partial s}^{(n+1)}$ to denote the class label which $x_s^{(n)}$ depends on, this time, ∂s is a set containing only one pixel at scale $n+1$. This simple dependency among class labels is often referred to as the quadtree model [12, 41], and its 1-D analog is shown in Figure 2.5. We further reduce the computation by assuming that each of the four children have the same probability distribution. Therefore, we replace the four distinct distributions used in the pyramidal graph model with a single distribution. We will denote the probability

mass function for each child by $\theta_{k,m,n} = p_{x_s^{(n)}|x_{\partial s}^{(n+1)}}(k|m)$ where $0 \leq k, m < M$ and $0 \leq n < L$. Since $\theta_{k,m,n}$ has at most M^2 distinct values for each scale n, we will use a look up table to represent this probability distribution.

In Appendix A, we use these assumptions to derive the following formula for computing the log likelihood terms

$$l_s^{(0)}(k) = \log p_{\tilde{y}_s^{(0)}|x_s^{(0)}}(\tilde{y}_s^{(0)}|k) \tag{2.10}$$

$$l_s^{(n)}(k) = \log p_{\tilde{y}_s^{(n)}|x_s^{(n)}}(\tilde{y}_s^{(n)}|k) + \sum_{i=1}^{4} \log \left\{ \sum_{m=0}^{M-1} \exp\left[l_{s_i}^{(n-1)}(m)\right] \theta_{m,k,n-1} \right\} \tag{2.11}$$

where s_i $(i = 1, 2, 3, 4)$ are the four children of s. Using (2.10) and (2.11), the log likelihood terms can be computed using a fine-to-coarse recursion through scales. First, the log likelihood term at the finest scale, $n = 0$, is calculated by applying equation (2.10). Then the log likelihood at the next coarser scale is computed with (2.11) for $n = 1$. This process is repeated until the coarsest scale is reached.

In our model, the feature vector at each pixel y_s is formed using the coefficients of a Haar basis wavelet decomposition. While the Haar basis is not very smooth, it is very computationally efficient to implement and does a good job of extracting useful feature vectors. The wavelet transform results in three bands at each resolution, which are often referred to at the low-high, high-low, and high-high bands. Because of the structure of the wavelet transform, each of these bands has half the spatial resolution of the original image. Each feature vector $y_s^{(n)}$ in our pyramid is then a three dimensional vector containing components from each of these three bands extracted at the same position in the image. Using this structure, the finest resolution of the pyramid has only half the resolution of the original image.

The conditional probability distribution of the feature vector's prediction error, $p_{\tilde{y}_s^{(n)}|x_s^{(n)}}(\cdot|k)$ can be modeled using a variety of statistical methods. In our approach, we use the multivariate Gaussian mixture model [44]

$$p_{\tilde{y}_s^{(n)}|x_s^{(n)}}(\tilde{y}|k) = \sum_{j=1}^{J_{k,n}} \gamma_{j,k,n} \frac{1}{(2\pi)^{3/2}|C_{j,k,n}|^{1/2}} \exp\left[-\frac{1}{2}(\tilde{y} - \mu_{j,k,n})^t C_{j,k,n}^{-1}(\tilde{y} - \mu_{j,k,n})\right] \tag{2.12}$$

where $J_{k,n}$ is the order of the Gaussian mixture for class k and scale n; and $\mu_{j,k,n}$, $C_{j,k,n}$, and $\gamma_{j,k,n}$ are the mean, covariance matrix, and weighting associated with the j-th component of the Gaussian mixture for class k and scale n. In general, $C_{j,k,n}$ will be positive definite, and $\gamma_{j,k,n} \in [0,1]$ with $\sum_{j=1}^{J_{k,n}} \gamma_{j,k,n} = 1$. For large $J_{k,n}$, the Gaussian mixture density can approximate any probability density.

2.4 Parameter Estimation

The SMAP segmentation algorithm described above depends on the selection of a variety of parameters that control the modeling of both data features and the context model. This section will explain how these parameters may be efficiently estimated from training data. The training data consists of a set of images together with their correct segmentations at the finest scale. This training data is then used to model both the texture characteristics and contextual structure of each region. The training process is performed in four steps:

1. Estimate of quadtree model parameters $\theta_{m,k,n}$ used in equation (2.11).

2. Decimate (subsample) the ground truth segmentations to form ground truth at all scales.

3. Estimate the Gaussian mixture model parameters of (2.12).

4. Estimate the coarse-to-fine transition probabilities $p_i^{(n)}(c|f)$ used in equation (2.8) by building an optimized class probability tree (CPT).

Perhaps the most important and difficult part of parameter estimation is step 4. This step estimates the parameters of the context model by observing the coarse-to-fine transition rates in the training data. Step 4 is a difficult incomplete data problem because we do not have access to the unknown class labels $X^{(n)}$ at all scales. One simple solution would be to estimate $p_i^{(n)}(c|f)$ from the subsampled ground truth labels computed in step 2. However, training from subsampled ground truth leads to biased estimates $p_i^{(n)}(c|f)$ that will result in excessive noise sensitivity in the SMAP

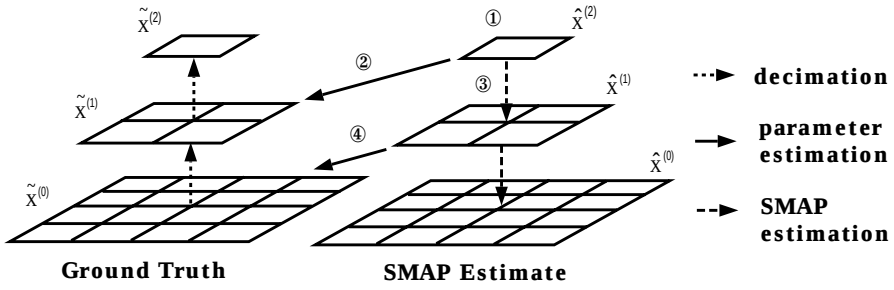


Fig. 2.6. Parameter estimation of the context model. (1) Compute the segmentation at the coarsest resolution, $\hat{x}^{(2)}$. (2) Estimate the transition probabilities $p_i^{(1)}(c|f)$ using the SMAP segmentation $\hat{x}^{(2)}$ and the decimated ground truth segmentation $\tilde{x}^{(1)}$. (3) Compute $\hat{x}^{(1)}$ using $p_i^{(1)}(c|f)$. (4) Estimate $p_i^{(0)}(c|f)$ using $\hat{x}^{(1)}$ and $\tilde{x}^{(0)}$. This procedure is then repeated for all scales.

segmentation. Alternatively, we have investigated the use of the EM algorithm together with Monte Carlo Markov chain techniques to compute unbiased estimates of the parameters [40]. While this methodology works, it is very computationally expensive and impractical for use with large sets of training data.

Our solution to step 4 is a novel coarse-to-fine estimation procedure which is computationally efficient and non-iterative, but results in accurate parameter estimates. The details of our method are explained in the following section 2.4.1.

Estimation of quadtree model parameters is discussed in section 2.4.2. The resulting quadtree model is then used to decimate the ground truth segmentation, so that ground truth is available at all scales. The resulting ground truth is then used to estimate Gaussian mixture model parameters using a well known clustering approach based on the EM algorithm.

2.4.1 Estimation of Context Model Parameters

Our context model is parameterized by the transition probabilities $p_i^{(n)}(c|f)$. Here f is a binary encoding of the coarse scale neighbors $X_{\partial s}^{(n)}$, and c is a binary encoding of the unknown pixel $X_s^{(n)}$. Notice that a different transition distribution is separately estimated for each scale, n, and for each of the four children i. This is important since it allows the model to be both scale and orientation dependent.

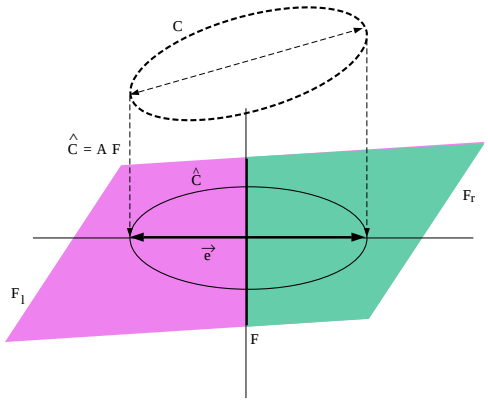


Fig. 2.7. Splitting rule based on the least squares estimation. The dash ellipse represents the covariance matrix of C and the solid ellipse represents the covariance matrix of $\hat{C}$, where $\hat{C}$ is the least squares estimate of C. $\vec{e}$ is the principle axis of the covariance matrix of $\hat{C}$. F is split into F_r and F_l according to the axis perpendicular to $\vec{e}$.

Our procedure for estimating the transition probabilities $p_i^{(n)}(c|f)$ is illustrated in Figure 2.6. The method works by estimating the transition probabilities from the coarser scale SMAP segmentation $\hat{x}^{(n+1)}$ to the correct ground truth segmentation denoted by $\tilde{x}^{(n)}$. Importantly, $\hat{x}^{(n+1)}$ does not depend on the transition probabilities $p_i^{(n)}(c|f)$. This can be seen from (2.4), the equation for computing the SMAP segmentation. This is a crucial fact since it allows $\hat{x}^{(n+1)}$ to be computed before $p_i^{(n)}(c|f)$ is estimated. Once $p_i^{(n)}(c|f)$ is estimated, it is then used to compute $\hat{x}^{(n)}$, allowing the estimation of $p_i^{(n-1)}(c|f)$. This process is then recursively repeated until the transition parameters at all scales are estimated.

In our approach, class probability trees are used to represent $p_i^{(n)}(c|f)$, so the ground truth $\tilde{x}^{(n)}$ and segmentation $\hat{x}^{(n+1)}$ will be used to construct and train the tree at each scale n and for each of the four child pixels $i = 1, 2, 3, 4$. We design the tree using the recursive tree construction (RTC) algorithm proposed by Gelfand, Ravishankar, and Delp [39], together with a multivariate splitting rule based on the least squares estimation. We have found that this method is very robust and yields tree depths that produce accurate segmentations. Determining the proper tree depth is very important because a tree that is too deep will over parameterize the model,

but a tree that is too shallow will not properly characterize the contextual structure of the training data.

The RTC algorithm works by partitioning the sample set into two halves. Initially, a tree is grown using the first partition, and then the tree is pruned using the second partition. Next the roles of the two partitions are swapped, with the second partition used for growing and the first partition used for pruning. This process is repeated, with partitions alternating roles, until the tree converges. At each iteration, the tree is pruned to minimize the misclassification probability on the data partition not being used for growing the tree.

In order to use the RTC algorithm, we must choose a method for growing the tree. Tree growing is done using a recursive splitting method. This method, illustrated in Figure 2.7, is based on a multivariate splitting procedure. First, the coarse scale neighbors, f, are used to compute $\hat{c}$, the least squares estimate of c. Then the values of $\hat{c}$ are split into two sets about the mean and along the direction of the principal eigenvector. The multivariate nature of the splitting procedure is very important because it allows clusters of f to be separated out efficiently.

More specifically, let t be the node being split into two nodes. We will assume that N samples of the training data pass into node t, so each sample of training data consists of the desired class label, c_n, and the coarse scale neighbors, f_n where $n = 1, \cdots, N$. Both c_n and f_n are binary encoded column vectors. Let μ_c and μ_f be the sample means for the two vectors

$$\begin{aligned} \mu_c &= \frac{1}{N}\sum_{n=1}^{N} c_n \\ \mu_f &= \frac{1}{N}\sum_{n=1}^{N} f_n \end{aligned}$$

We may then define the matrices

$$\begin{aligned} C &= [c_1 - \mu_c, c_2 - \mu_c, \ldots, c_N - \mu_c] \\ F &= [f_1 - \mu_f, f_2 - \mu_f, \ldots, f_N - \mu_f] \end{aligned}$$

The least squares estimate of C given F is then

$$\hat{C} = [CF^t(FF^t)^{-1}]F \ .$$

Let $\vec{e}$ be the principal eigenvector of the covariance matrix $R = \hat{C}\,\hat{C}^t$. Then our splitting rule is : if $A_t f - \mu_t \geq 0$, f goes to the left child of t; otherwise, f goes to the right child of t, where

$$\begin{aligned} A_t &= \vec{e}^t CF^t(FF^t)^{-1} \\ \mu_t &= A_t\,\mu_f \ . \end{aligned}$$

At each step, we split the node which results in the largest decrease in entropy for the tree. This is done by splitting all the candidate nodes in advance and computing the entropy reduction for each node.

2.4.2 Estimation of Quadtree Parameters

The quadtree model is parameterized by the transition probabilities

$$p_{x_s^{(n)}|x_{\partial s}^{(n+1)}}(k|m) = \theta_{k,m,n}$$

, where $x_s^{(n)} = k$ and $x_{\partial s}^{(n+1)} = m$. As with the context model parameters, estimation of the parameters $\theta_{k,m,n}$ is an incomplete data problem because the true segmentation classes are not known at each scale. However, in this case the EM algorithm [45] can be used to solve this problem in a computationally efficient way.

For our problem, the EM algorithm can be written as the following iterative procedure.

$$\theta^{(j+1)} = \arg\max_{\theta} E\left[\log p(X^{(>0)}|\theta) \mid \tilde{x}^{(0)}, \theta^{(j)}\right] \tag{2.13}$$

where $\theta^{(j)}$ are the estimated quadtree parameters at iteration j, and $\tilde{x}^{(0)}$ is the ground truth segmentation at the finest resolution. Using our model, the maximization in (2.13) has the following solution.

$$\theta_{k,m,n}^{(j+1)} = \frac{\sigma_{k,m,n}^{(j)}}{\sum_{l=0}^{M-1} \sigma_{l,m,n}^{(j)}} \tag{2.14}$$

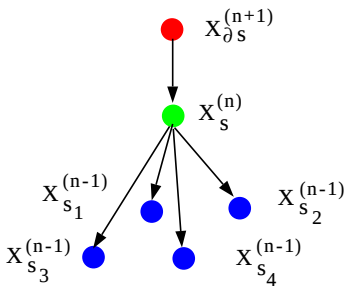


Fig. 2.8. Dependency among class labels in the quadtree model. Given class labels at all pixels except $x_s^{(n)}$, $x_s^{(n)}$ only depends on class labels of its parent, $x_{\partial s}^{(n+1)}$, and four children, $x_{s_i}^{(n-1)}$.

where $\sigma_{k,m,n}^{(j)}$ is defined as the following.

$$\sigma_{k,m,n}^{(j)} = \sum_{s \in S^{(n)}} p(x_s^{(n)} = k, x_{\partial s}^{(n+1)} = m \mid \tilde{x}^{(0)}, \theta^{(j)})$$

The conditional probabilities $p(x_s^{(n)} = k, x_{\partial s}^{(n+1)} = m \mid \tilde{x}^{(0)}, \theta^{(j)})$ can be computed using either a recursive formula [46, 47] or stochastic sampling techniques. The recursive formulations have the advantage of giving exact update expressions for (2.13). However, we have found that for this application stochastic sampling methods are easily implemented and work well.

The stochastic sampling approach requires two steps. First, samples of $X^{(>0)}$ are generated using the Gibbs sampler [48]. Then, $\sigma_{k,m,n}^{(j)}$ is estimated using the histogram of the samples. For the quadtree model, the Gibbs sampler can be easily implemented, because the class label of a pixel, $x_s^{(n)}$ only depends on the class label of its parent $x_{\partial s}^{(n+1)}$ and the class labels of its four children $x_{s_i}^{(n-1)}$ (see Figure 2.8). The detailed algorithm for stochastic sampling is given in Appendix B.

2.4.3 Decimation of Ground Truth Segmentation

After the quadtree models are estimated, we will use them to decimate the fine resolution ground truth to form ground truth segmentations at all resolutions. Importantly, simple decimation algorithms do not give the best results. For example, simple majority voting tends to smear or remove fine details of a segmentation. Figure 2.9(a) is a ground truth segmentation, and the decimated segmentations using

the majority voting are shown in Figure 2.9(b). Clearly, most of the fine details, such as text lines, and captions are removed by repeated decimation. To address this problem, we will use a decimation algorithm based on the maximum likelihood (ML) estimation. Figure 2.9(c) shows the results using our ML approach. Notice that the fine details are well preserved in Figure 2.9(c).

Our ML estimate of the ground truth at scale n is given by

$$\tilde{x}^{(n)} = \arg\max_{x^{(n)}} p_{\tilde{x}^{(0)}|x^{(n)}}(\tilde{x}^{(0)}|x^{(n)}) \ .$$

This can be easily computed by first computing log likelihood terms in a fine-to-coarse recursion as in equations (2.10) and (2.11).

$$\begin{aligned} \tilde{l}_s^{(1)}(k) &= \sum_{i=1}^{4} \log \theta_{\tilde{x}_{s_i}^{(0)},k,0} \\ \tilde{l}_s^{(n)}(k) &= \sum_{i=1}^{4} \log \left\{ \sum_{m=0}^{M-1} \exp\left[\tilde{l}_{s_i}^{(n-1)}(m)\right] \theta_{m,k,n-1} \right\} \end{aligned}$$

and then selecting the class label which maximizes the log likelihood at each pixel.

$$\tilde{x}_s^{(n)} = \arg \max_{0 \le k \le M-1} \tilde{l}_s^{(n)}(k)$$

2.4.4 Estimation of Data Model Parameters

In section 2.3.2, we have used the Gaussian mixture model of equation (2.12) to approximate the conditional probability distribution $p_{\tilde{y}_s^{(n)}|x_s^{(n)}}(\tilde{y}|k)$. The EM algorithm is a standard algorithm for estimating parameters of a mixture model [44, 45]. We use the EM algorithm to estimate the means $\mu_{j,k,n}$, the covariance matrices $C_{j,k,n}$, and the weights $\gamma_{j,k,n}$ for each Gaussian mixture density. The model order $J_{k,n}$ is chosen for each class k using the Rissanen criteria [49]. Training data set are generated using the feature vectors $y^{(n)}$ and ground truth segmentation $\tilde{x}^{(n)}$. The prediction coefficients defined in (2.9) are estimated from training data using the standard least squares estimation.

2.5 Experimental Results

In this section, we apply our segmentation algorithm to the problem of document segmentation. Document segmentation is a interesting test case for the algorithm

because documents have complex contextual structure which can be exploited to improve segmentation accuracy. In addition, multiscale features are important for documents since regions such as text, picture, and background can only be accurately distinguished by using texture features at both small and large scales. For a review of document segmentation algorithms, one can refer to [4]. To distinguish our algorithm from the SMAP algorithm proposed in [12], we will call our algorithm the trainable SMAP (TSMAP) algorithm.

The TSMAP algorithm is tested on a database of 50 grayscale document images scanned at 100dpi on an low cost 32 bit flat-bed scanner. We use the scanned images as they are with no pre-processing. In some cases, the images contain "ghosting" artifacts when images and text on the back of a document image can "bleed through" during the scanning process. The database of 50 images was partitioned into 20 training images and 30 testing images. Each of the 20 training images was manually segmented into three classes: text, picture and background. These segmentations were then used as ground truth for parameter estimation. Four training images and their associated ground truth segmentations are shown in Figure 2.10.

In our experiments, we allowed a maximum of 8 resolution levels where level 0 is the finest resolution, and level 7 is the coarsest. For each resolution, prediction errors were modeled using the Gaussian mixture model discussed in section 2.3.2. Each Gaussian mixture density contained 15 or fewer mixture components. Unless otherwise stated, a 5×5 coarse neighborhood was used. We found that this neighborhood size gave the best overall performance while minimizing computation. For all our segmentation results, we use "red", "green", and "blue" to represent text, picture and background regions respectively.

Figure 2.11 illustrates the segmentation of a document image in the testing set. Figure 2.11(a) is the original image, Figure 2.11(b) shows the result of segmentation using the proposed segmentation algorithm, referred as TSMAP algorithm, with a 5×5 coarse scale neighborhood, Figure 2.11(c) shows the segmentation using TSMAP with a 1×1 coarse scale neighborhood, and Figure 2.11(d) shows the segmentation

using only the finest resolution features combined with the Markov random field as the context model. Figures 2.12-2.13 show the segmentation results for another 6 images outside the training set using TSMAP segmentation with a 5×5 neighborhood.

Notice that the larger 5×5 neighborhood substantially improves the accuracy of segmentation when compared to the 1×1 neighborhood. This is because the large neighborhood can more accurately account for large scale contextual structure in the image. For the 5×5 neighborhood, the "picture" regions are enforced to be uniform, while "text" regions are allowed to be small with fine detail. Even single text lines, reverse text (white text on dark background) and page numbers are correctly segmented. The algorithm also works robustly in the presences of different types of background. For example, white paper and halftoned color background have different textual behavior, but the model allows them to both be handled correctly. The result produced using a MRF prior model is much poorer. This is not surprising since the parameters of the prior model can not be adapted to the document structure. Regions between text lines are frequently misclassified and edges of the picture regions are quite irregular.

Figure 2.14 shows the effect of the training set size on the quality of the resulting segmentation. The TSMAP algorithm with a 5×5 coarse scale neighborhood is trained on three training sets which consist of 20, 10, 5 training images, respectively. The resulting segmentations are shown in Figure 2.14(c)-(h). Notice that the segmentation quality degrades as the number of training images is decreased, but that good results are obtained with as few as 10 training images. However, when the number of training images is too small, such as 5, the segmentation results (see Figure 2.14(g)-(h)) can become unreliable.

2.6 Conclusion

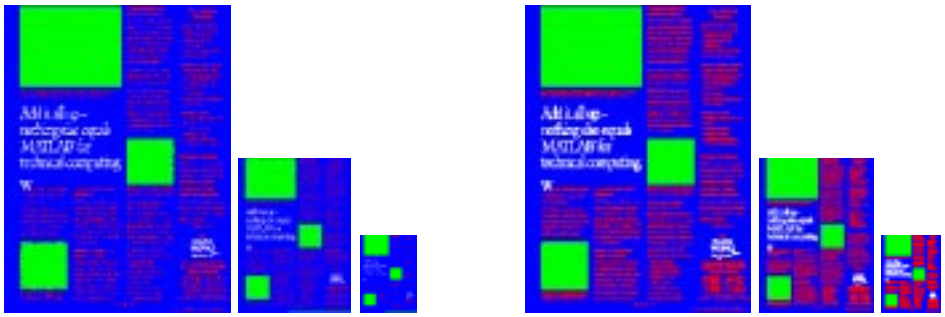
We proposed a new approach to multiscale Bayesian image segmentation which allows for accurate modeling of complex contextual structure. The method uses a Markov chain in scale to model both the texture features and the contextual dependencies for the image. In order to capture the complex dependencies, we use a class

probability tree to model the transition probabilities of the Markov chain. The class probability tree allows us to use a large neighborhood of dependencies while simultaneously limiting the number of parameters that must be estimated. We also propose a novel training technique which allows the context model parameters to be efficiently estimated in a noniterative coarse-to-fine procedure.

In order to test our algorithm, we apply it to the problem of document segmentation. This problem is interesting both because of its practical significance and because the contextual structure of documents is complex. Experiments with scanned document images indicate that the new approach is computationally efficient and improves the segmentation accuracy over fixed scale Bayesian segmentation methods.

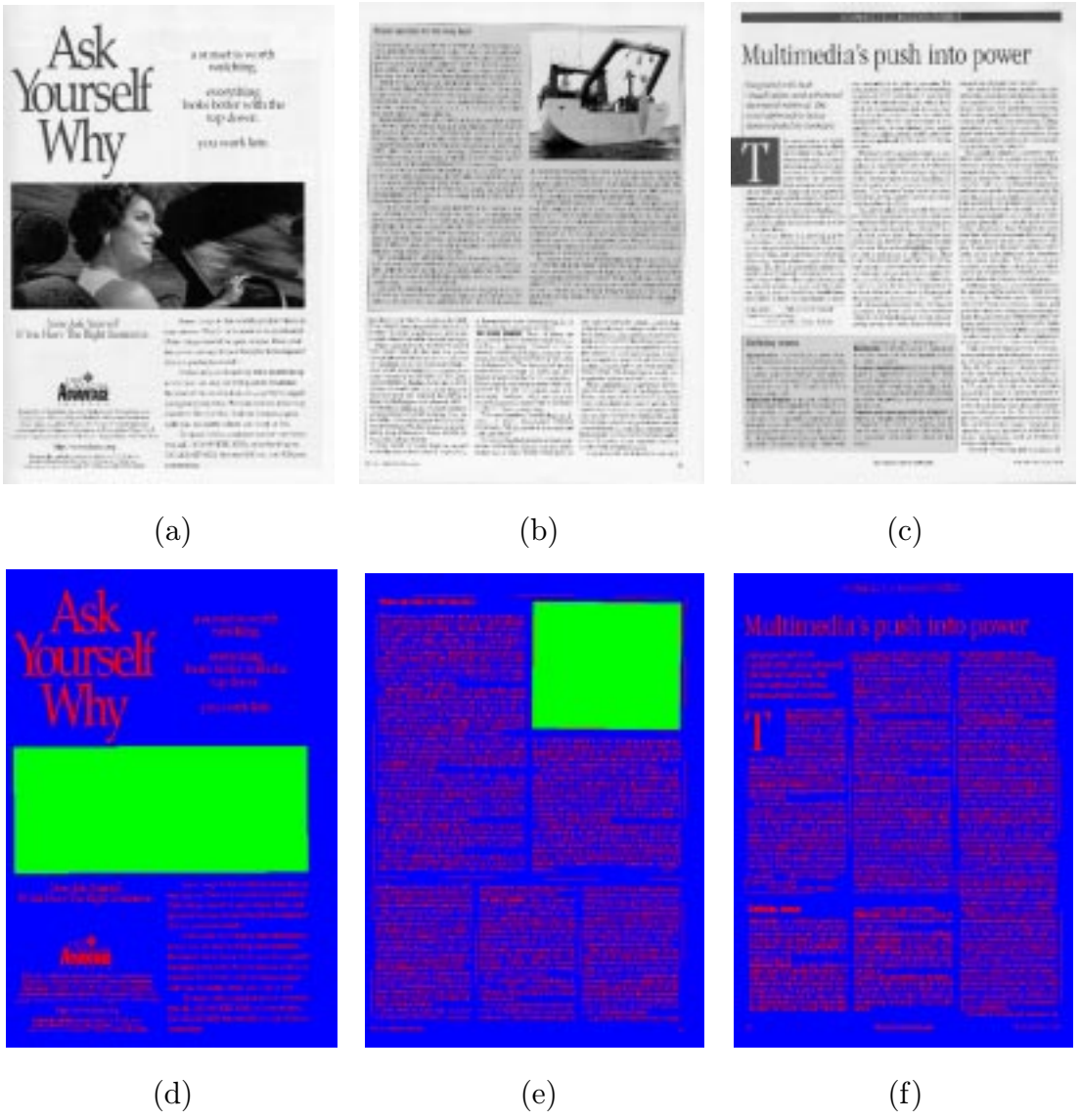


(a)



(b) (c)

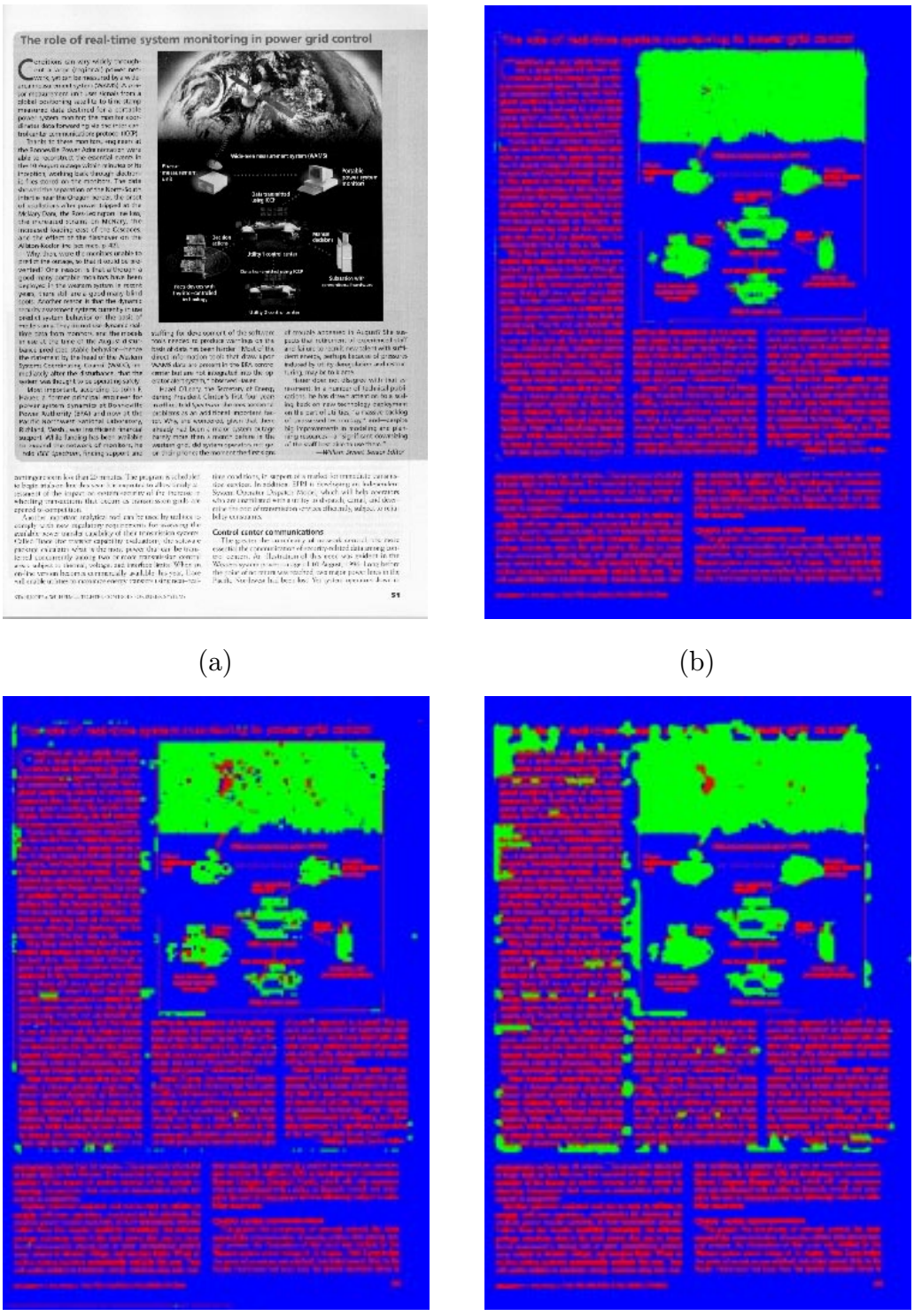
Fig. 2.9. The ground truth image and decimated ground truth images for n=0,1,2. (a) Ground truth segmentation. (b) Decimated ground truth segmentations using majority voting. (c) Decimated ground truth segmentations using ML estimate.



(a) (b) (c)

(d) (e) (f)

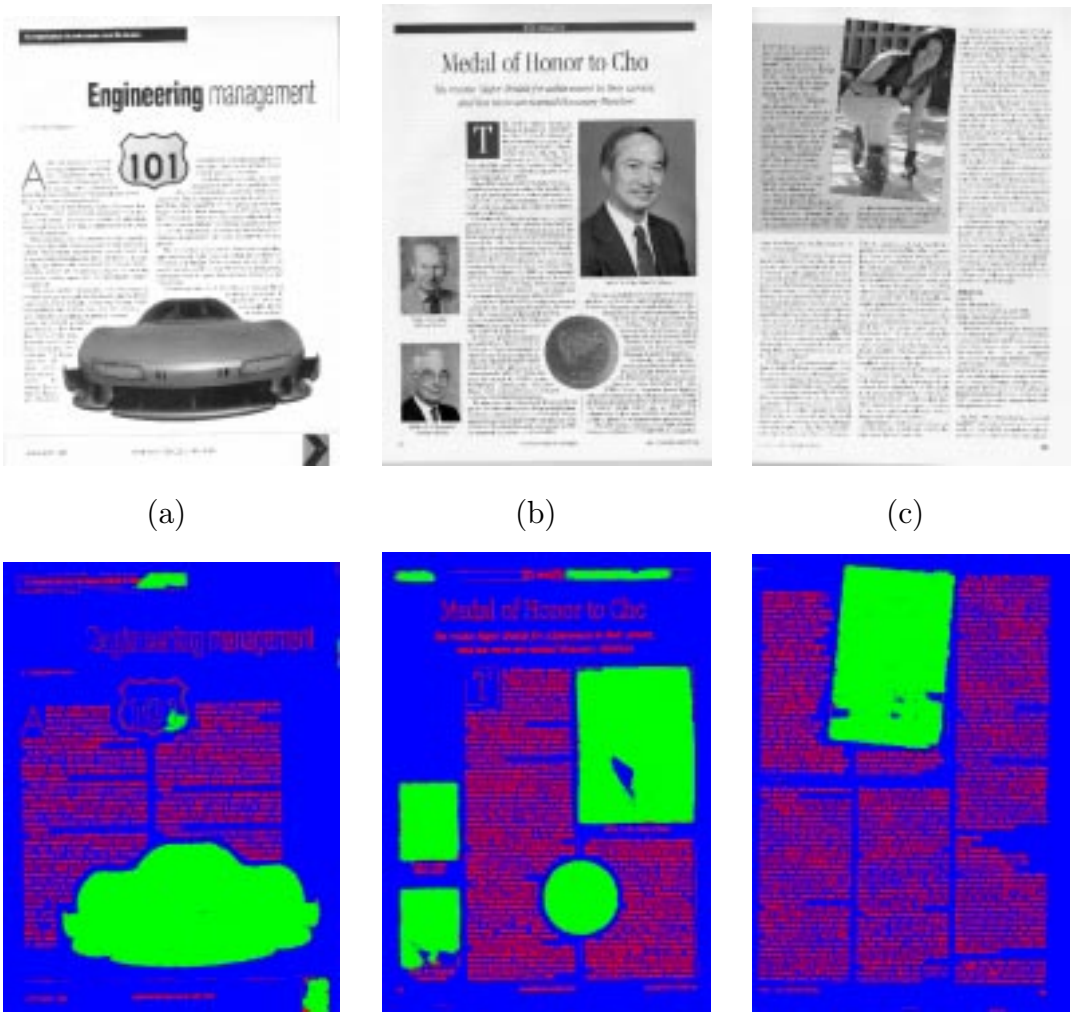
Fig. 2.10. Training images and their corresponding ground truth segmentations: (a)-(c) are training images, and (d)-(f) are ground truth segmentations. Red, green, blue represent text, picture, and background, respectively.



(a) (b)

(c) (d)

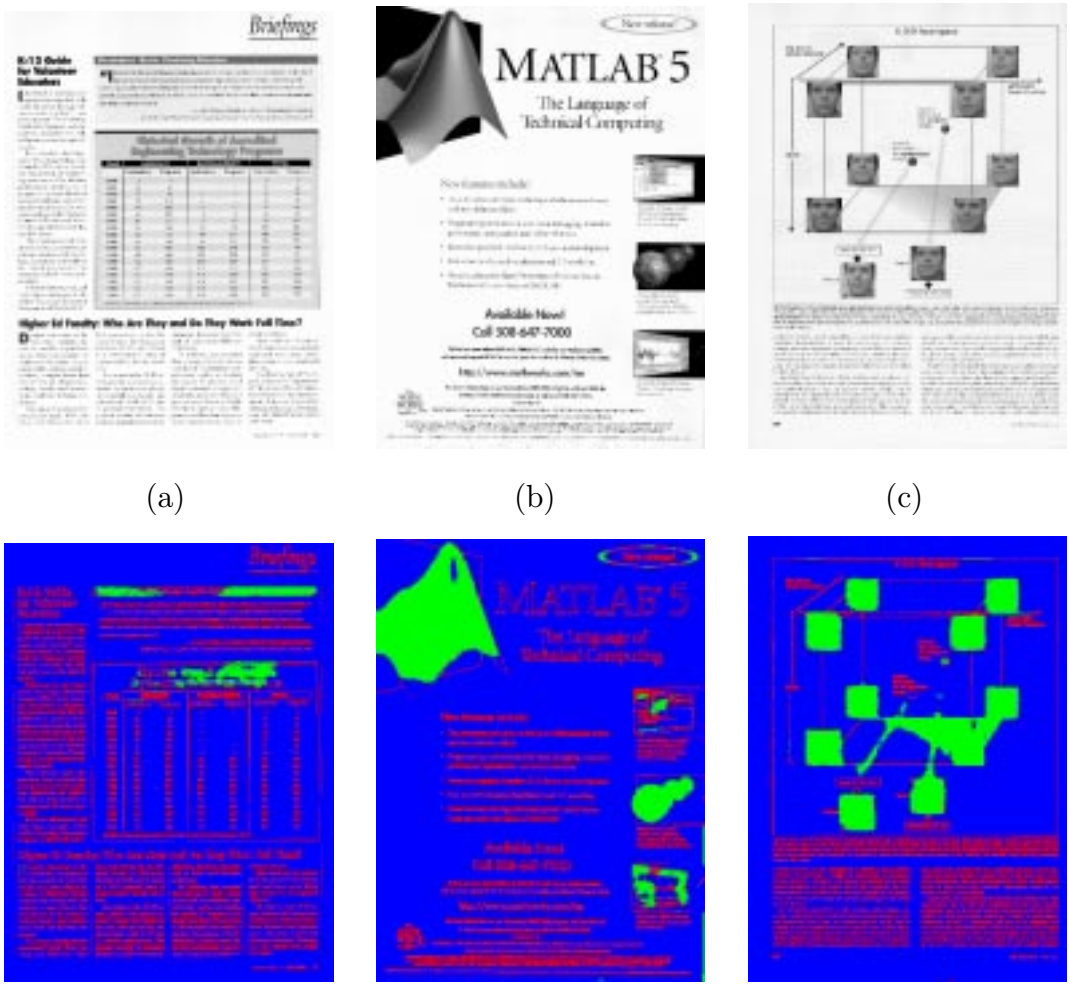
Fig. 2.11. Comparison of segmentation results among different algorithms: (a) Original image. (b) Segmentation result using TSMAP with a 5×5 neighborhood. (c) Segmentation result using TSMAP with a 1×1 neighborhood. (d) Segmentation result using Markov random field. Red, green and blue represent text, picture and background, respectively.



(a) (b) (c)

(d) (e) (f)

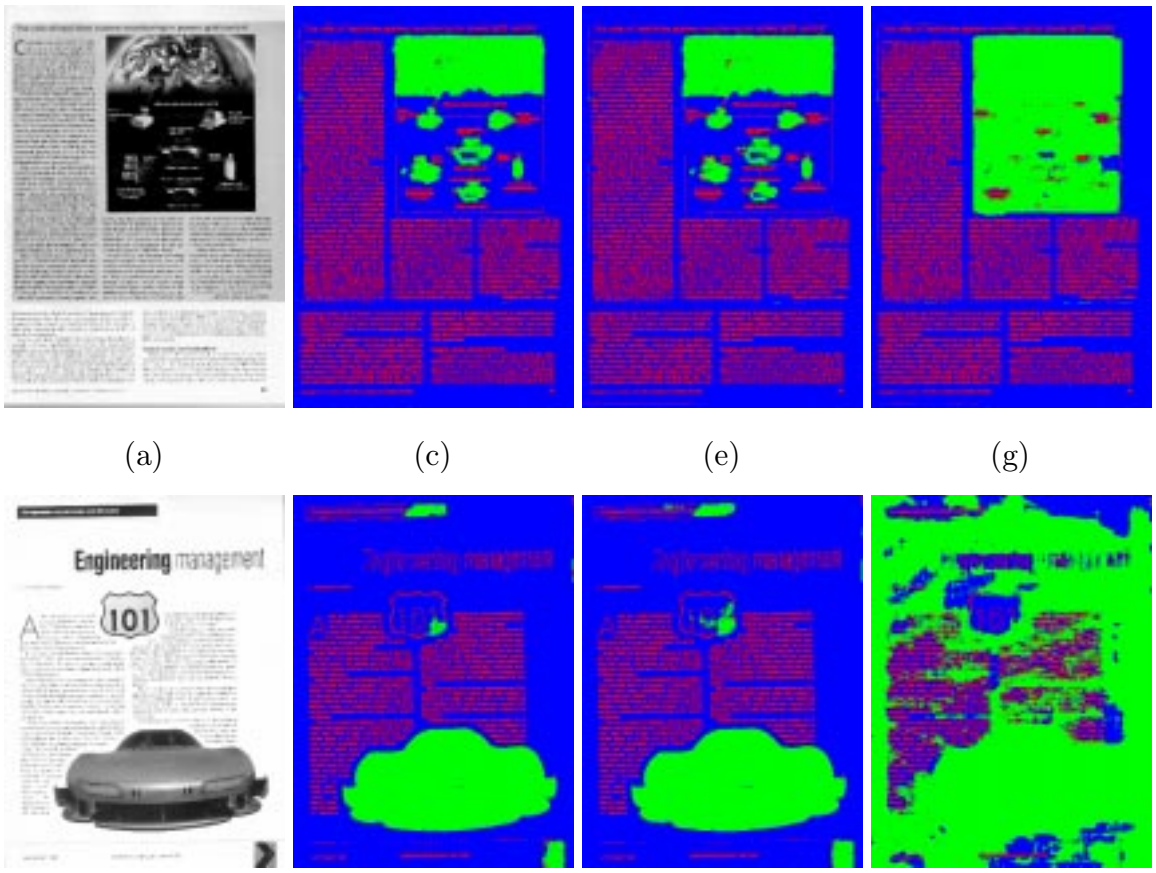
Fig. 2.12. TSMAP Segmentation results I: (a)-(c) Original images. (d)-(f) Segmentation results using TSMAP with a 5×5 neighborhood. Red, green, and blue represent text, picture and background, respectively.



(a) (b) (c)

(d) (e) (f)

Fig. 2.13. TSMAP segmentation results II: (a)-(c) Original images. (d)-(f) Segmentation results using TSMAP with a 5×5 neighborhood for 4 different test images. Red, green, blue represent text, picture, and background respectively.



(a) (c) (e) (g)

(b) (d) (f) (h)

Fig. 2.14. The effect of the number of training images on TSMAP: (a)-(b) Original images. (c)-(d) TSMAP segmentation results when trained on 20 images. (e)-(f) TSMAP segmentation results when trained on 10 images. (g)-(h) TSMAP segmentation results when trained on 5 images. For all cases, a 5×5 coarse neighborhood is used. Red, green and blue represent text, picture and background, respectively.

3. Document Compression Using Rate-Distortion Optimized Segmentation

Effective document compression algorithms require that scanned document images be first segmented into regions such as text, pictures and background. In this chapter, we introduce a multilayer compression algorithm for document images. This compression algorithm first segments a scanned document image into different classes, then compresses each class using an algorithm specifically designed for that class. Also, we propose a rate-distortion optimized segmentation (RDOS) algorithm designed to work with document compression. The RDOS algorithm works in a closed loop fashion by applying each coding method to each region of the document and then selecting the method that yields the best rate-distortion trade-off. Compared with the TSMAP algorithm, the RDOS algorithm can often result in a better rate-distortion trade-off, and produce more robust segmentations by eliminating those misclassifications which can cause severe artifacts. At similar bit rates, the multilayer compression algorithm using RDOS can achieve a much higher subjective quality than state-of-the-art compression algorithms, such as DjVu and SPIHT.

3.1 Introduction

Common office devices such as digital photocopiers, fax machines, and scanners require that paper documents be digitally scanned, stored, transmitted and then printed or displayed. Typically, these operations must be performed rapidly, and user expectations of quality are very high since the final printed output is often subject to close inspection. Digital implementation of this imaging pipeline is particularly formidable when one considers that a single page of a color document scanned at 400-600 dpi (dots per inch) requires approximately 45-100 Megabytes of storage. Consequently, practical systems for processing color documents require document compression methods

that achieve high compression ratios and with very low distortion.

Document images differ from natural images because they usually contain well defined regions with distinct characteristics, such as text, line graphics, continuous-tone pictures, halftone pictures and background. Typically, text requires high spatial resolution for legibility, but does not require high color resolution. On the other hand, continuous-tone pictures need high color resolution, but can tolerate low spatial resolution. Therefore, a good document compression algorithm must be spatially adaptive, in order to meet different needs and exploit different types of redundancy among different image classes. Traditional compression algorithms, such as JPEG, are based on the assumption that the input image is spatially homogeneous, so they tend to perform poorly on document images.

Most existing compression algorithms for document images can be roughly classified as block-based approaches and layer-based approaches. Block-based approaches, such as [5, 50, 6, 8], segment non-overlapping blocks of pixels into different classes, and compress each class differently according to its characteristics. On the other hand, layer-based approaches [51, 52, 7, 53] partition a document image into different layers, such as the background layer and the foreground layer. Then, each layer is coded as an image independent from other layers. Most layer-based approaches use the three-layer (foreground/mask/background) representation proposed in the ITU's Recommendations T.44 for mixed raster content (MRC). The foreground layer contains the color of text and line graphics, and the background layer contains pictures and background. The mask is a bi-level image which determines, for each pixel in the reconstructed image, if the foreground color or the background color should be used.

The performance of a document compression system is directly related to its segmentation algorithm. A good segmentation can not only lower the bit rate, but also lower the distortion. On the other hand, those artifacts which are most damaging are often caused by misclassifications.

Some segmentation algorithms which have been proposed for document compression use features extracted from the discrete cosine transform (DCT) coefficients to

separate text blocks from picture blocks. For example, Murata [5] proposed a method based on the absolute values of DCT coefficients, and Konstantinides and Tretter [6] use a DCT activity measure to switch among different scale factors of JPEG quantization matrices. Other segmentation algorithms are based on the features extracted directly from the document image. The DjVu document compression system [52] uses a multiscale bi-color clustering algorithm to separate foreground and background. In [7], text and line graphics are extracted from a check image using morphological filters followed by thresholding. Ramos and de Queiroz proposed a block-based activity measure as a feature for separating edge blocks, smooth blocks and detailed blocks for document coding [8].

In this chapter, we introduce a multilayer document compression algorithm. This algorithm first classifies 8×8 non-overlapping blocks of pixels into different classes, such as text, picture and background. Then, each class is compressed using an algorithm specifically designed for that class. Two segmentation algorithms are used for the multilayer compression algorithm: a direct image segmentation algorithm called the trainable sequential MAP (TSMAP) algorithm [41], and a rate-distortion optimized segmentation (RDOS) algorithm developed for document compression [54].

The TSMAP algorithm proposed in Chapter 2 is representative of most document segmentation algorithms in that it computes the segmentation from only the input document image. The disadvantage of such direct segmentation approaches for document coding is that they do not exploit knowledge of the operational performance of the individual coders, and that they can not be easily optimized for different target bit-rates.

In order to address these problems, we propose a segmentation algorithm which optimizes the actual rate-distortion performance for the image being coded. The RDOS method works by first applying each coding method to each region of the image, and then selecting the class for each region which approximately maximizes the rate-distortion performance. The RDOS optimization is based on the measured distortion and an estimate of the bit rate for each coding method. Compared with

direct image segmentation algorithms (such as the TSMAP segmentation algorithm), RDOS has several advantages. First, RDOS produces more robust segmentations. Intuitively, misclassifications which cause severe artifacts are eliminated because all possible coders are tested for each block of the image. In addition, RDOS allows us to control the trade-off between the bit rate and the distortion by adjusting a weight. For each weight set by a user, an approximately optimal segmentation is computed in the sense of rate and distortion.

Recently, there has been considerable interest in optimizing the operational rate-distortion characteristics of image coders. Ramchandran and Vetterli [55] proposed a rate-distortion optimal way to threshold or drop quantized DCT coefficients of a JPEG or an MPEG coder. Effros and Chou [56] introduced a two-stage bit allocation algorithm for a simple DCT-based source coder.[2] Their encoder uses a collection of quantization matrices, and each block of DCT coefficients is quantized using a quantization matrix selected by the "first-stage quantizer". The two-stage bit allocation is optimized in the sense of rate and distortion. Schuster and Katsaggelos [15] apply rate-distortion optimization for video coding. But importantly, they also model the 1-D inter-block dependency for estimating the bit rate and distortion, and the optimization problem is solved by dynamic programming techniques. For a comprehensive review of rate-distortion methods for image compression, one can refer to [57].

Our approach to optimizing rate-distortion performance differs from these previous methods in a number of important ways. First, we switch among different types of coders, rather then switching among sets of parameters for a fixed vector quantizer (VQ), DCT, or Karhunén-Loeve (KL) transform coder. In particular, we use a coder optimized for text representation that can not be represented as a DCT coder, VQ coder, or KL transform coder. Our text coder works by segmenting each block into foreground and background pixels in a manner similar to that used by Harrington and

[2]The DCT-base coder used in [56] differs from JPEG because the DC component is not differentially encoded, and no zigzag run-length encoding of the AC components is used.

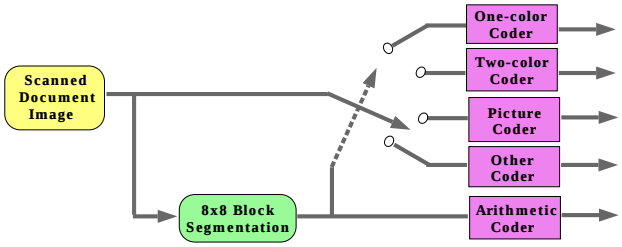


Fig. 3.1. General structure of the multilayer document compression algorithm.

Klassen [50]. By exploiting the bi-level nature of text, this coder gives performance which is far superior to what can be achieve with transform coders. Another distinction of our method is that the different coders use somewhat different distortion measures. This is motivated by the fact that perceived quality for text, graphics and pictures is different. A class-dependent distortion measure is also found valuable in [8].

We test the multilayer compression algorithm on both scanned and noiseless synthetic document images. For typical document images, we can achieve compression ratios ranging from 180:1 to 250:1 with very high quality reconstructions. In addition, experimental results show that, in this range of compression ratios, the multilayer compression algorithm using RDOS results in a much higher subjective quality than well-known compression algorithms, such as DjVu, SPIHT [58] and JPEG.

3.2 Multilayer Compression Algorithm

The multilayer compression algorithm shown in Fig. 3.1 classifies each 8×8 block of pixels into one of four possible classes: Picture block, Two-color block, One-color block, and Other block. Each of the four classes corresponds to a specific coding algorithm which is optimized for that class. The class labels of all blocks are compressed and sent as side information.

The flow diagram of our compression algorithm is shown in Fig. 3.2. Ideally, One-color blocks should be from uniform background regions, and each One-color block is represented by an indexed color. The color indices of One-color blocks are finally entropy coded using an arithmetic coder. Two-color blocks are from text or line

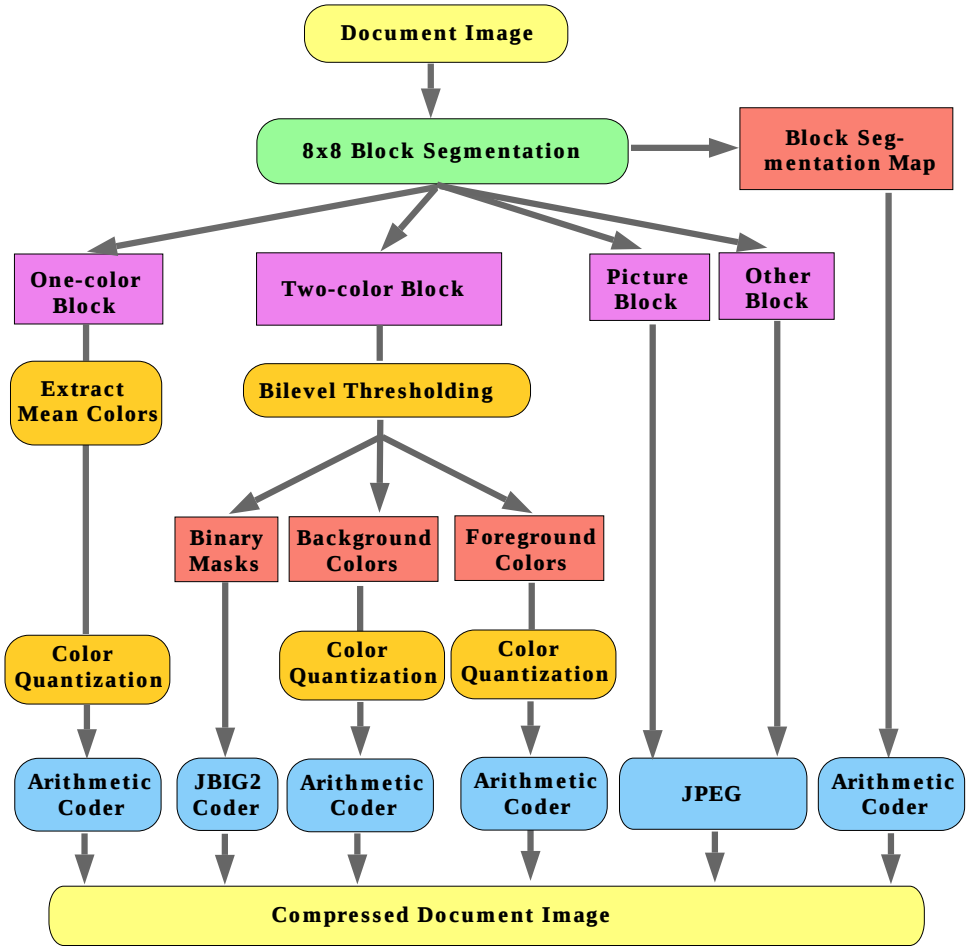


Fig. 3.2. Flow diagram of the multilayer document compression algorithm.

graphics, and they need to be coded with high spatial resolution. Therefore, for each Two-color block, a bi-level thresholding is used to extract two colors (one foreground color and one background color) and a binary mask. Since Two-color blocks can tolerate low color resolution, both the foreground and the background colors of Two-color blocks are first quantized, and then entropy coded using an arithmetic coder. The binary masks are coded using a JBIG2 coder. Picture blocks are generally from regions containing either continuous-tone or halftone picture data, these blocks are compressed by JPEG using customized quantization tables. In addition, some regions of text and line graphics can not be accurately represented by Two-color blocks. For example, thin lines bordered by regions of two different colors require a minimum

of three or more colors for accurate representation. We assign these problematic blocks to the Other block class. Other blocks are JPEG compressed together with Picture blocks. But they use different quantization tables which have much lower quantization steps than those used for Picture blocks. The details of compression and decompression of each of these four classes are described in the following subsections.

Throughout this chapter, we use y to denote the original image and x to denote its 8×8 block segmentation. Also, y_i denotes the i-th 8×8 block in the image, where the blocks are taken in raster order, and x_i denotes the class label of block i, where $0 \leq i < L$, and L is the total number of blocks. The set of class labels is then $\mathcal{N} = \{One, Two, Pic, Oth\}$, where One, Two, Pic, Oth represent One-color, Two-color, Picture, and Other blocks, respectively.

3.2.1 Compression of One-color Blocks

Each One-color block is represented by an indexed color. Therefore, for One-color blocks, we first extract the mean color of each block, and then color quantize the mean colors of all One-color blocks. Finally, the color indices are entropy coded using a third order arithmetic coder [59]. When reconstructing One-color blocks, smoothing is used among adjacent One-color blocks if their maximal difference along all three color coordinates is less than 12.

3.2.2 Compression of Two-color Blocks

The Two-color class is designed to compress blocks which can be represented by two colors, such as text blocks. Since Two-color blocks need to be coded with high spatial resolution, but can tolerate low color resolution, each Two-color block is represented by two indexed colors and a binary mask. The bi-level thresholding algorithm that we use for extracting the two colors and the binary mask uses a minimal mean squared error (MSE) thresholding followed by a spatially adaptive refinement. The algorithm is performed on two block sizes. First, 8×8 blocks are used. But sometimes an 8×8 block may not contain enough samples from both color regions for a reliable estimate of the colors of both regions and the binary mask. In this case, a 16×16 block centered at the 8×8 block will be used instead.

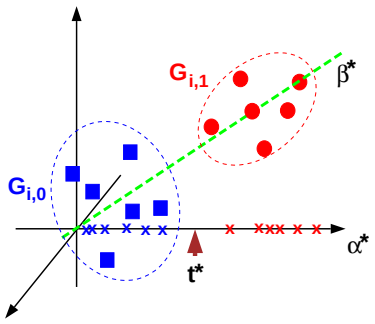


Fig. 3.3. Minimal MSE thresholding. We use α^* to denote the color axis with the largest variance, and β^* to denote the principle axis. t^* is the optimal threshold on α^*, and x's are the samples projected on α^*.

The minimal MSE thresholding algorithm is illustrated in Fig. 3.3. For a Two-color block y_i, we first project all colors of y_i onto the color axis α^* which has the largest variance among three color axes. The thresholding is done only on α^*. Since we are mainly interested in high quality document images where text is sharp and the noise level is low, the projection step significantly lowers the computation complexity without sacrificing the quality of the bi-level thresholding. For a threshold t on α^*, t partitions all colors into two groups. Let $E_i(t)$ be the MSE, when colors in each group are represented by the mean color of that group. We compute the value t^* which minimizes $E_i(t)$. Then, t^* partitions the block into two groups, $G_{i,0}$ and $G_{i,1}$, where the mean color of $G_{i,0}$ has a larger l_1 norm than the mean color of $G_{i,1}$. Let $c_{i,j}$ be the mean color of $G_{i,j}$, where $j = 0, 1$. Then, $\|c_{i,0}\|_1 > \|c_{i,1}\|_1$ is true for all i. We call $c_{i,0}$ the background color of block i, and $c_{i,1}$ the foreground color of block i. The binary mask which indicates the locations of $G_{i,0}$ and $G_{i,1}$ is denoted as $b_{i,m,n}$, where $b_{i,m,n} \in \{0, 1\}$, and $0 \leq m, n \leq 7$.

The minimal MSE thresholding usually produces a good binary mask. But $c_{i,0}$ and $c_{i,1}$ are often biased estimates. This is mainly caused by the boundary points between two color regions since their colors are a combination of the colors of the two regions. Therefore, $c_{i,0}$ and $c_{i,1}$ need to be refined. Let a point in block i be an internal point of $G_{i,j}$, if the point and its 8-nearest neighbors all belong to $G_{i,j}$. If a

point is not an internal point of either $G_{i,0}$ or $G_{i,1}$, we call it a boundary point. Also, denote the set of internal points of $G_{i,j}$ as $\tilde{G}_{i,j}$. If $\tilde{G}_{i,j}$ is not empty, we set $c_{i,j}$ to the mean color of $\tilde{G}_{i,j}$. When $\tilde{G}_{i,j}$ is empty, we can not estimate $c_{i,j}$ reliably. In this case, if the current block size is 8×8, we will enlarge the block to 16×16 symmetrically along all directions, and use the same algorithm to extract two colors and a 16×16 mask. Then, the two colors extracted from the 16×16 block are used as $c_{i,0}$ and $c_{i,1}$, and middle portion of the 16×16 mask is used as $b_{i,m,n}$. If $\tilde{G}_{i,j}$ is empty, and the current block size is a 16×16 block, $c_{i,j}$ will be used as it is without refinement.

After bi-level thresholding, foreground colors, $\{c_{i,1}|x_i = Two\}$, and background colors, $\{c_{i,0}|x_i = Two\}$, of all Two-color blocks are quantized separately. Then, the color indices of foreground colors are packed in raster order, and compressed using a third order arithmetic coder. So are the color indices of background colors.

To compress the binary masks, $b_{i,m,n}$, we form them into a single binary image B which has the same size as the original document image y. Any block in B which *does not* correspond to a Two-color block is set to 0's, and any block corresponding to a Two-color block is set to the appropriate binary mask $b_{i,m,n}$. The binary image B is then compressed by a JBIG2 coder using the lossless soft pattern matching technique [60].

3.2.3 Compression of Picture Blocks and Other Blocks

Picture blocks and Other blocks are all compressed using JPEG. Therefore, they are also called JPEG blocks. Picture blocks are compressed using a quantization tables similar to the standard JPEG quantization table at quality level 20; however, the quantization steps for the DC coefficients in both luminance and chrominance are set to 15. Other blocks use the standard JPEG quantization tables at quality level 75.

The JPEG standard generally uses 2×2 subsampling of the two chrominance channels to reduce the overall bit rate. This means that each 8×8 JPEG chrominance block will correspond to four JPEG blocks in the luminance channel. If any one of the four luminance blocks is JPEG'ed, then the corresponding chrominance block will

also be JPEG'ed. More specifically, the class of each chrominance block is denoted by z_j, where j indexes the block. The class of the chrominance block can take on the values $z_j \in \{Pic, Oth, NoJ\}$, where NoJ indicates that the chrominance block is not JPEG'ed. The specific choice of z_j will depend on the choice of either the TSMAP and RDOS methods of segmentation and will be discussed in detail in sections 3.2.5 and 3.3.

All the JPEG luminance blocks (i.e. those of type Pic or Oth) are packed in raster order, and then JPEG coded using conventional zigzag run length encoding followed by the default JPEG Huffman entropy coding. The same procedure is used for the chrominance blocks of type Pic or Oth but with the corresponding chrominance JPEG default Huffman table. We note that the number of luminance blocks will in general be less then four times the number of chrominance blocks. This is because some chrominance blocks may correspond to a set of four luminance blocks that are not all JPEG'ed. As an implementational detail, we pad these missing luminance blocks with zeros so that we can use the standard JPEG library routines provided by the Independent JPEG Group.

3.2.4 Additional Issues

The block segmentation x for the luminance blocks is entropy coded using a third order arithmetic coder. We will see that for the TSMAP method, the chrominance block segmentation, z, can be computed from x, so it does not need to be coded separately. However, for the RDOS method, $z = \{z_j\}$ is also entropy coded with a third order arithmetic coder.

As stated above, the Two-color blocks and One-color blocks use color quantization as a preprocessing step to coding. Color quantization vector quantizes the set of colors into a relatively small set or palette. Importantly, different classes use different color palettes for the quantization since this improves the quality without significantly increasing the bit rate. In all cases, we use the binary splitting algorithm of [61] to perform color quantization. The binary splitting algorithm is terminated when either the number of colors exceeds 255 or the principal eigenvalue of the covariance matrix

of every leaf node is less then a threshold of 10 for the One-color blocks and 30 for the Two-color blocks.

3.2.5 Use of the TSMAP Segmentation Algorithm

To use the multilayer compression algorithm, a document image needs first to be segmented. In this section, we will discuss how to use the TSMAP segmentation algorithm proposed in Chapter 2 in the multilayer compression algorithm.

For a document image, we first use the TSMAP algorithm to segment each block into One-color, Two-color or Picture blocks. Other blocks are then selected from Two-color blocks using a post processing operation. Recall from section 3.2.2 that each Two-color block y_i, is partitioned into two groups $G_{i,0}$ and $G_{i,1}$. Then, we calculate the average distance (in YCrCb color space) of the boundary points to the line determined by $\tilde{c}_{i,0}$ and $\tilde{c}_{0,1}$, where $\tilde{c}_{i,0}$ is the quantized background color and $\tilde{c}_{i,1}$ is the quantized foreground color. If the average distance is larger than 45, re-classify the current block to Other block. Also, if the total number of internal points of $G_{i,0}$ and $G_{i,1}$ is less than or equal to 8, we re-classify the current block to One-color block.

When TSMAP is used, the class of each chrominance block is determined from the classes of the four corresponding luminance blocks.

If any of the four luminance blocks is of type Oth,
  then set chrominance block to Oth.
Else if any of the four luminance blocks is of type Pic,
  then set chrominance block to Pic.
Else set chrominance block to NoJ.

Intuitively, each chrominance block is set to the highest quality of its corresponding luminance blocks.

The current implementation of the TSMAP algorithm can only be used for grayscale images. In addition, because the structure of the wavelet decomposition used for feature extraction, TSMAP produces a segmentation map which has half the spatial resolution of the input image. Therefore, in order to compute an 8×8 block segmen-

tation of a 400 dpi color image, we first subsample the original image by a factor of 4 using block averaging, and then convert the subsampled image into a grayscale image. The grayscale image will be used as the input image to TSMAP for computing the 8×8 block segmentation.

3.3 Rate-Distortion Optimized Segmentation

In this section, we will discuss a rate distortion optimized segmentation (RDOS) method designed for use with the multilayer document compression algorithm. The RDOS method works in a closed loop fashion by applying each coder to each region of the document and then selecting the coder that yields the best rate-distortion trade-off.

In order to better understand the role of segmentation in document compression, we will first compare two different types of segmentation algorithms: the trainable sequential MAP (TSMAP) algorithm of [41] proposed in Chapter 2, and the RDOS algorithm described in this section. The TSMAP is representative of a broad class of direct segmentation algorithms that segment the document based solely on the document image. In essence, the TSMAP method makes decisions without regard to the specific properties or performance of the individual coders that are used. Its advantage is simplicity since it does require that each coding method be applied to each region of the document. However, we will see that direct segmentation methods, such as TSMAP, have two major disadvantages. First, they tend to result in infrequent but serious misclassification errors. For example, even if only a few Two-color blocks are misclassified as One-color blocks, these misclassifications will lead to broken lines and smeared text strokes that can severely degrade the quality of the document. Second, the segmentation is usually computed independently of the bit rate and the quality desired by the user. This causes inefficient use of bits and even artifacts in the reconstructed image.

Alternatively, the RDOS method requires greater computation, but insures that each block is coded using the method which is best suited to it. We will see that this results in more robust segmentations which yield a better rate-distortion trade-off at

every quality level.

Let $R(y|x)$ be the number of bits required to code y with block segmentation x. Let $R(x)$ be the number of bits required to code x, and let $D(y|x)$ be the total distortion resulting from coding y with segmentation x. Then, the rate-distortion optimized segmentation, x^*, is

$$x^* = \arg \min_{x \in \mathcal{N}^L} \{R(y|x) + R(x) + \lambda D(y|x)\}, \tag{3.1}$$

where λ is a non-negative real number which controls the trade-off between bit rate and distortion. In our approach, we assume that λ is a constant controlled by a user which has the same function as the quality level in JPEG.

To compute RDOS, we need to estimate the number of bits required for coding each block using each coder, and the distortion of coding each block using each coder. For computational efficiency, we assume that the number of bites required for coding a block only depends on the image data, and class labels of that block and the previous block in raster order. We also assume that the distortion of a block can be computed independently from other blocks. With these assumptions, (3.1) can be rewritten as

$$x^* = \arg \min_{\{x_0, x_1, \ldots, x_{L-1}\} \in \mathcal{N}^L} \sum_{i=0}^{L-1} \{R_i(x_i|x_{i-1}) + R_x(x_i|x_{i-1}) + \lambda D_i(x_i)\}, \tag{3.2}$$

where $R_i(x_i|x_{i-1})$ is the number of bits required to code block i using class x_i given x_{i-1}, $R_x(x_i|x_{i-1})$ is the number of bits needed to code the class label of block i, and $D_i(x_i)$ is the distortion produced by coding block i as class x_i. After the rate and distortion are estimated for each block and each coder, (3.2) can be solved using a dynamic programming technique similar to that used in [15].

An important aspect of our approach is that we use a class-dependent distortion measure. This is desirable because, for document images, different regions, such as text, background and pictures, can tolerate different types of distortion. For example, errors in high frequency bands can be ignored in background and picture regions, but they can cause severe artifacts in text regions.

In the following sections, we specify how to compute the rate and distortion terms for each of the four classes, One-color, Two-color, Picture and Other. The expres-

sions for rate are often approximate due to the difficulties of accurately modeling high performance coding methods such as JBIG2. However, our experimental results indicate that these approximations are accurate enough to consistently achieve good compression results. For the purposes of this work, we also assume that the term $R_x(x_i|x_{i-1}) = 0$. This is reasonable after coding the block segmentation x requires only an insignificant number of overhead bits, typically less then 0.01 bits per color pixel.

3.3.1 Estimate Bit Rates and Distortion of One-color Blocks

Recall from section 3.2.1 that each One-color block is represented by an indexed color. Color indices of all One-color blocks are entropy coded with a third order arithmetic coder. But for simplicity, the number of bits used for coding a One-color block is estimated with a first order approximation. That is when x_i and x_{i-1} are all One-color blocks, we let

$$R_i(x_i|x_{i-1}) = -\log_2 p_\mu(\mu_i|\mu_{i-1}),$$

where μ_i is the indexed color of block i, and $p_\mu(\mu_i|\mu_{i-1})$ is the transition probability of indexed colors between adjacent blocks. When x_{i-1} is not a One-color block, we let

$$R_i(x_i|x_{i-1}) = -\log_2 p_\mu(\mu_i).$$

To estimate $p_\mu(\mu_i|\mu_{i-1})$ and $p_\mu(\mu_i)$, we assume that all blocks are One-color blocks, and compute the probabilities.

In addition, the total squared error in YCrCb color space is used as the distortion measure of One-color blocks. If $x_i = One$, then

$$D_i(x_i) = \sum_{m=0}^{7}\sum_{n=0}^{7} \|y_{i,m,n} - \mu_i\|^2,$$

where $y_{i,m,n}$ is the color of pixel (m, n) in the i-th block y_i, $0 \le m, n \le 7$, and $\|a\| = \sqrt{a^t a}$.

3.3.2 Estimate Bit Rates and Distortion of Two-color Blocks

A Two-color block is represented by two indexed colors and a binary mask. For block i, let $\tilde{c}_{i,0}$, $\tilde{c}_{i,1}$ be the two indexed colors, and let $b_{i,m,n}$ be the binary mask for

block i where $0 \leq m, n \leq 7$. Then, in the reconstructed image, the color of pixel (m, n) in block i is $\tilde{c}_{i,b_{i,m,n}}$.

The bits used for coding the two indexed colors are approximated as

$$-\sum_{j=0}^{1} \log_2 p_j(\tilde{c}_{i,j}|\tilde{c}_{i-1,j}) \,,$$

where $p_j(\tilde{c}_{i,j}|\tilde{c}_{i-1,j})$ is the transition probability of the j-th indexed color between adjacent blocks in raster order. We also assume that the number of bits for coding $b_{i,m,n}$ only depends on its four causal nearest neighbors, denoted as

$$V = [b_{i,m-1,n-1}, b_{i,m-1,n}, b_{i,m-1,n+1}, b_{i,m,n-1}]^t.$$

Define $b_{i,m,n}$ to be 0, if $m < 0$ or $n < 0$ or $m > 7$ or $n > 7$. Then, the number of bits required to code the binary mask is approximated as

$$-\sum_{m=0}^{7}\sum_{n=0}^{7} \log_2 p_b(b_{i,m,n}|V_{i,m,n}),$$

where $p_b(b_{i,m,n}|V_{i,m,n})$ is the transition probability from the four causal nearest neighbors to pixel (m, n) in block i. Therefore, when x_i and x_{i-1} are all Two-color blocks, the total number of bits is estimated as

$$R_i(x_i|x_{i-1}) = -\sum_{j=0}^{1} \log_2 p_j(\tilde{c}_{i,j}|\tilde{c}_{i-1,j}) - \sum_{m=0}^{7}\sum_{n=0}^{7} \log_2 p_b(b_{i,m,n}|V_{i,m,n}).$$

If x_{i-1} is not a Two-Color block, we use $p_j(\tilde{c}_{i,j})$ instead of $p_j(\tilde{c}_{i,j}|\tilde{c}_{i-1,j})$ to estimate the number of bits for coding the color indices. The probabilities $p_j(\tilde{c}_{i,j})$, $p_j(\tilde{c}_{i,j}|\tilde{c}_{i-1,j})$ and $p_b(b_{i,m,n}|V_{i,m,n})$ are estimated for all 8×8 blocks whose maximal dynamic range along the three color axes is larger or equal to 8.

The distortion measure used for Two-color blocks is designed with the following considerations. In a scanned image, pixels on the boundary of two color regions tend to have a color which is a combination of the colors of both regions. Since only two colors are used for the block, the boundaries between the color regions are usually sharpened. Although the sharpening generally improves the quality, it gives a large difference in pixel values between the original and the reconstructed images on

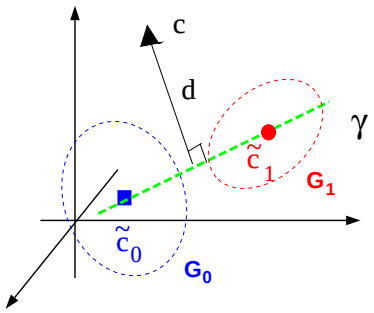


Fig. 3.4. Two-color distortion measure. $\tilde{c}_0$ and $\tilde{c}_1$ are indexed mean colors of group G_0 and G_1, respectively. γ is the line determined by $\tilde{c}_0$ and $\tilde{c}_1$. The distance between a color c and γ is d. When c is a combination of $\tilde{c}_0$ and $\tilde{c}_1$, $d = 0$.

boundary points. On the other hand, if a block is not a Two-color block, a third color often appears on the boundary. Therefore, a desired distortion measure for Two-color coder should not excessively penalize the error caused by sharpening, but has to produce a high distortion value, if more than two colors exist. Also, desired Two-color blocks should have a certain proportion of internal points. If a Two-color block has very few internal points, the block usually comes from background or halftone background, and it can not be a Two-color block. To handle this case, we set the cost to the maximal cost, if the number of internal points is less than or equals to 8.

The distortion measure for the Two-color block is defined as follows. We also define $I_{i,m,n}$ as an indicator function. $I_{i,m,n} = 1$, if (m, n) is an internal point. $I_{i,m,n} = 0$, if (m, n) is a boundary point. If $x_i = Two$,

$$D_i(x_i) = \begin{cases} \sum_{m=0}^{7}\sum_{n=0}^{7}\left[I_{i,m,n}\|y_{i,m,n} - \tilde{c}_{i,b_{i,m,n}}\|^2 \right. \\ \qquad \left. +(1 - I_{i,m,n})d^2(y_{i,m,n}; \tilde{c}_{i,0}, \tilde{c}_{i,1})\right], & if \sum_{j=0}^{1}|\tilde{G}_{i,j}| > 8 \\ 255^2 \times 64 \times 3, & if \sum_{j=0}^{1}|\tilde{G}_{i,j}| \le 8 \end{cases}$$

where $|\tilde{G}_{i,j}|$ is the number of elements in the set $\tilde{G}_{i,j}$, and $d(y_{i,m,n}; \tilde{c}_{i,0}, \tilde{c}_{i,1})$ is the distance between $y_{i,m,n}$ and the line determined by $\tilde{c}_{i,0}$ and $\tilde{c}_{i,1}$. As illustrated in Fig. 3.4, if a color c is a combination of c_1 and c_2, c will be on the line determined

by c_1 and c_2, $d(c; c_1, c_2) = 0$. Therefore, for boundary points of Two-color blocks, $d(y_{i,m,n}; \tilde{c}_{i,0}, \tilde{c}_{i,1})$ is small. However, if a third color does exist on a boundary point, $d(y_{i,m,n}; \tilde{c}_{i,0}, \tilde{c}_{i,1})$ tends to be large.

3.3.3 Estimate Bit Rates and Distortion of JPEG Blocks

JPEG blocks contain both Picture blocks and Other blocks. The bits required for coding a JPEG block i can be divided into two parts: the bits required for coding the luminance of block i, denoted as $R_i^l(x_i|x_{i-1})$, and the bits for coding the chrominance, denoted as $R_i^c(x_i|x_{i-1})$. Therefore,

$$R_i(x_i|x_{i-1}) = R_i^l(x_i|x_{i-1}) + R_i^c(x_i|x_{i-1}).$$

Let $\alpha_i^d(x_i)$ be the quantized DC coefficients of the luminance using the quantization table specified by class x_i, and $\alpha_i^a(x_i)$ be the vector which contains all 63 quantized AC coefficients of the luminance of block i. Using the standard Huffman tables, $R_i^l(x_i|x_{i-1})$ can be computed as

$$R_i^l(x_i|x_{i-1}) = r_d\left[\alpha_i^d(x_i) - \alpha_{i-1}^d(x_{i-1})\right] + r_a\left[\alpha_i^a(x_i)\right],$$

where $r_d(\cdot)$ is the number of bits used for coding the difference between two consecutive DC coefficients of the luminance component, and $r_a(\cdot)$ is the number of bits used for coding AC coefficients. The formula for calculating $r_d(\cdot)$ and $r_a(\cdot)$ is specified in the JPEG standard [62]. Notice that when x_{i-1} is also a JPEG class, $R_i(x_i|x_{i-1})$ is the exact number of bits required for coding the luminance component using JPEG. If x_{i-1} is not a JPEG class, we assume that the previous quantized DC value is 0. (In the JPEG library, a 0 DC value corresponds to a block average of 128.)

Since the two chrominance components are subsampled 2×2, we approximate the number of bits for coding the chrominance components of an 8×8 block i, $R_i^c(x_i|x_{i-1})$, as follows. Let j be the index of the 16×16 block which contains block i. Also, let $\beta_{j,k}^d(x_i)$ be the quantized DC coefficient of the k-th chrominance component using the chrominance quantization table of class x_i, and $\beta_{j,k}^a(x_i)$ be the vector of the quantized

AC coefficients. Then, we assume that

$$R_i^c(x_i|x_{i-1}) = \frac{1}{4}\sum_{k=0}^{1}\left\{r'_d\left[\beta^d_{j,k}(x_i) - \beta^d_{j-1,k}(x_i)\right] + r'_a\left[\beta^a_{j,k}(x_i)\right]\right\},$$

where $r'_d(\cdot)$ is the number of bits used for coding the difference between two consecutive DC coefficients of the chrominance components, and $r'_a(\cdot)$ is the number of bits used for coding AC coefficients of the chrominance components. Notice that we split the bits used for coding the chrominance equally among the four corresponding 8×8 blocks of the original image, and assume that the classes of the chrominance blocks j and $j-1$ are all x_i.

The total squared error in YCrCb is used as the distortion measure for JPEG blocks. The distortion is computed in the DCT domain, eliminating the need to compute inverse DCT's. Let $\tilde{\alpha}_i$ be the un-quantized DCT coefficients of the luminance component of block i, and $\tilde{\beta}_{j,k}$ be the un-quantized DCT coefficients of the k-th chrominance component of the 16×16 block containing block i. Then, the distortion is approximately given by

$$D_i(x_i) = \left\|\tilde{\alpha}_i - \alpha_i(x_i)\right\|^2 + \sum_{k=0}^{1}\left\|\tilde{\beta}_{j,k} - \beta_{j,k}(x_i)\right\|^2.$$

Here, we approximate the distortion due to the chrominance channels by dividing the chrominance error among the four corresponding 8×8 blocks of the luminance channel.

In RDOS, the chrominance segmentation is not computed from the 8×8 block segmentation x. It is computed separately using a similar rate-distortion approach followed by a post-processing step. Let $\tilde{y}_j$ be the j-th 16×16 block in raster order. We first compute a 16×16 block segmentation $z = \{z_0, z_1, \ldots, z_{L/4-1}\}$ which is rate-distortion optimized using the constrain that $z \in \{Pic, Oth\}^{L/4}$. Ignoring the bits used for coding z, z is computed as

$$z = \arg\min_{z' \in \{Pic,Oth\}^{L/4}} \sum_{j=0}^{L/4-1}\left\{\tilde{R}_j(z'_j|z'_{j-1}) + \lambda\tilde{D}_j(z'_j)\right\},$$

where $\tilde{R}_j(z_j|z_{j-1})$ is the number of bits required for coding $\tilde{y}_j$ with segmentation z_i

given z_{j-1},

$$\tilde{R}_j(z_j|z_{j-1}) = \sum_{k=0}^{1} \left\{ r'_d \left[\beta^d_{j,k}(z_j) - \beta^d_{j-1,k}(z_{j-1}) \right] + r'_a \left[\beta^a_{j,k}(z_j) \right] \right\}$$

and $\tilde{D}_j(z_j)$ is the distortion of coding $\tilde{y}_j$ with segmentation z_j.

$$\tilde{D}_j(z_j) = \sum_{k=0}^{1} \left\| \tilde{\beta}_{j,k} - \beta_{j,k}(z_j) \right\|^2$$

Finally, in the post-processing step, we set z_j to NoJ, if none of the four 8×8 blocks corresponding to j is either a Picture block or an Other block.

3.4 Experimental Results

For our experiments, we use an image database consisting of 30 scanned and one synthetic document image. The scanned documents come from a variety of sources, including ASEE Prism and IEEE Spectrum. These documents are scanned at 400 dpi and 24 bits per pixel (bpp) using the HP flat-bed scanner, scanjet 6100C. A large portion of the 30 scanned images contain halftone background and have ghosting artifacts caused by printing on the reverse side of the page. These images are used without pre-processing. The synthetic image shown in Fig. 3.9 has a complex layout structure and many colors. It is used to test the ability of a compression algorithm to handle complex document images. The TSMAP segmentations are computed using the parameters obtained in [41]. These parameters were extracted from a separate set of 20 manually segmented grayscale images scanned at 100 dpi.

Fig. 3.5(a) and (d) show the original test image I and test image II [1]. Their TSMAP segmentations are shown in Fig. 3.5(b) and (e). Fig. 3.5(c) is the RDOS segmentation of test image I with $\lambda = 0.0021$, and Fig. 3.5(c) is the RDOS segmentation of test image II with $\lambda = 0.0018$. The bit rates and compression ratios of these test images compressed by the multilayer compression algorithm using both TSMAP and RDOS are shown in Table 3.1.

Both TSMAP and RDOS segmentations classify most of the regions correctly. In many ways, TSMAP segmentations appear better than RDOS segmentations with

[1] ©1994 IEEE. Reprinted, with permission, from IEEE Spectrum, page 33, July 1994.

image	segmentation algorithm	bit rate (bbp)	compression ratio	RDOS distortion per pixel per color	λ
Test image I	TSMAP	0.138	173:1	27.58	n/a
	RDOS	0.132	182:1	23.47	0.0021
	RDOS	0.125	192:1	24.99	0.0018
	RDOS	0.095	253:1	31.00	0.0013
Test image II	TSMAP	0.120	200:1	40.33	n/a
	RDOS	0.114	210:1	32.14	0.0018
Test image III (Synthetic)	TSMAP	0.089	245:1	32.12	n/a
	RDOS	0.101	237:1	3.40	0.0042

Table 3.1
Bit rates, compression ratios and RDOS distortion per pixel per color channel of three test images compressed by the multilayer compression algorithm using both TSMAP and RDOS.

solid picture regions and clearly defined boundaries. In contrast, the RDOS segmentation often classifies smooth regions of pictures as One-color class. In fact, this yields a lower bit rate without producing noticeable distortion. More importantly, RDOS more accurately segments Two-color blocks. For example, in Fig. 3.5 (e), several line segments in the graphics are misclassified as One-color blocks.

In Fig. 3.6, we compare the quality of reconstructed images compressed using both the TSMAP segmentation and the RDOS segmentation at similar bit rates. Figures 3.6(a), (b) and (c) show a portion of test image I together with the results of compression using the TSMAP and RDOS methods. We can see from Fig. 3.6(b) that several text strokes are smeared, when the image is compressed using the TSMAP segmentation. These artifacts are caused by misclassifying Two-color blocks as One-color blocks. This type of misclassification does not occurred in the RDOS segmentation.

In Table 3.2, we list the average bit rate and standard deviation of coding each

classes	average bit rate (bbp)	standard deviation
One-color	0.0240	0.0092
Two-color	0.3442	0.1471
JPEG	0.8517	0.3260
Segmentations	0.0097	0.0002

Table 3.2
Mean and standard deviation of the bit rate of coding each class computed over 30 document images scanned at 400 dpi and 24 bpp. These images are compressed using RDOS with $\lambda = 0.0018$.

class computed over 30 scanned document images. These images are compressed using RDOS segmentation with $\lambda = 0.0018$. Although JPEG classes include Picture class and Other class, when $\lambda = 0.0018$, very few blocks are segmented as Other blocks. Therefore, the listed average bit rate for JPEG classes is close to the average bit rate for Picture class. The bit rate for segmentations includes both for the 8×8 block segmentation and the chrominance segmentation. For a document image, if the percentage of One-color, Two-color and JPEG blocks is known, we can estimate the bit rate of the image compressed by our algorithm using the average bit rate of each class.

Figure 3.7 shows the RDOS segmentations of test image I using different λ's, where $\lambda_1 = 0.013$ and $\lambda_2 = 0.018$. It can be seen that for smaller λ, less weight is put on the distortion, and more blocks are segmented as One-color blocks. When λ increases, more weight is put on the distortion, and more blocks are segmented as Picture blocks. But in all cases, text blocks are reliably classified as λ changes within a reasonable range.

In Fig. 3.8, we compare the rate-distortion performance achieve by the multi-layer compression algorithm using RDOS, TSMAP and manual segmentations. Figure 3.8(a) is computed from test image I shown in Fig. 3.5(a), and Fig. 3.8(b) is computed from test image III, the synthetic image shown in Fig. 3.9(a). The x-axis is

the bit rate, and the y-axis is the average distortion per pixel per color channel, where the distortion is defined in section 3.3. The solid lines in Fig. 3.8 are the true rate-distortion curves with RDOS, and the dash lines are the estimated rate-distortion curves with RDOS using both estimated bit rate and estimated distortion. It can be seen that the distortion is estimated quite accurately, but the bit rate tends to be over-estimated by a fixed constant. The manual segmentations are generated by an operator to achieve the best possible performance. Notice that for a document image with a simple layout, such as test image I, the manual segmentation has a comparable rate-distortion performance with the RDOS segmentation. However, for a document image with a complex layout, such as test image III, the manual segmentation shown in Fig. 3.9(c) has rate-distortion performance which is inferior to which is achieved by the RDOS segmentation. Both the RDOS and the manual segmentation have superior rate-distortion performance to TSMAP.

Figures 3.10–3.13 compare, at similar bit rates, the quality of the reconstructed images compressed using RDOS segmentation with those compressed using three well-known coders: DjVu [52], SPIHT [58], and JPEG. Among the three coders, DjVu is designed for compressing scanned document images. It uses the basic three-layer MRC model, where the foreground and the background are subsampled and compressed using a wavelet coder, and the bi-level mask is compressed using JBIG2. Since DjVu is designed to view and browse document images on the web, it can achieve very high compression ratios, but the quality of the reconstructed images tends not to be very high, especially for images with complex layouts and many color regions. SPIHT is a state-of-the-art wavelet coder. It works well for natural images, but it fails to compress document images at a low bit rate with high fidelity. For our test images, the baseline JPEG usually can not achieve the desired bit rate, around 0.1 bpp, at which the other three algorithms operate. Even at a bit rate near 0.2 bpp, JPEG still generates severe artifacts.

Figure 3.10 shows a comparison of the four algorithms for a small region of color text in test image III. The RDOS method clearly out-performs other algorithms on the

color text region. Fig. 3.11(a) is another part of test image III, where a logo is overlaid on a continuous-tone image. It is difficult to say whether this region should belong to Picture class or Two-color class. However, since RDOS uses a localized rate and distortion trade-off, it performs well in this region, producing a much sharper result than those coded using DjVu or SPIHT. A disadvantage of SPIHT is that many bits are used to code text regions, so it does not allocate enough bits for picture regions.

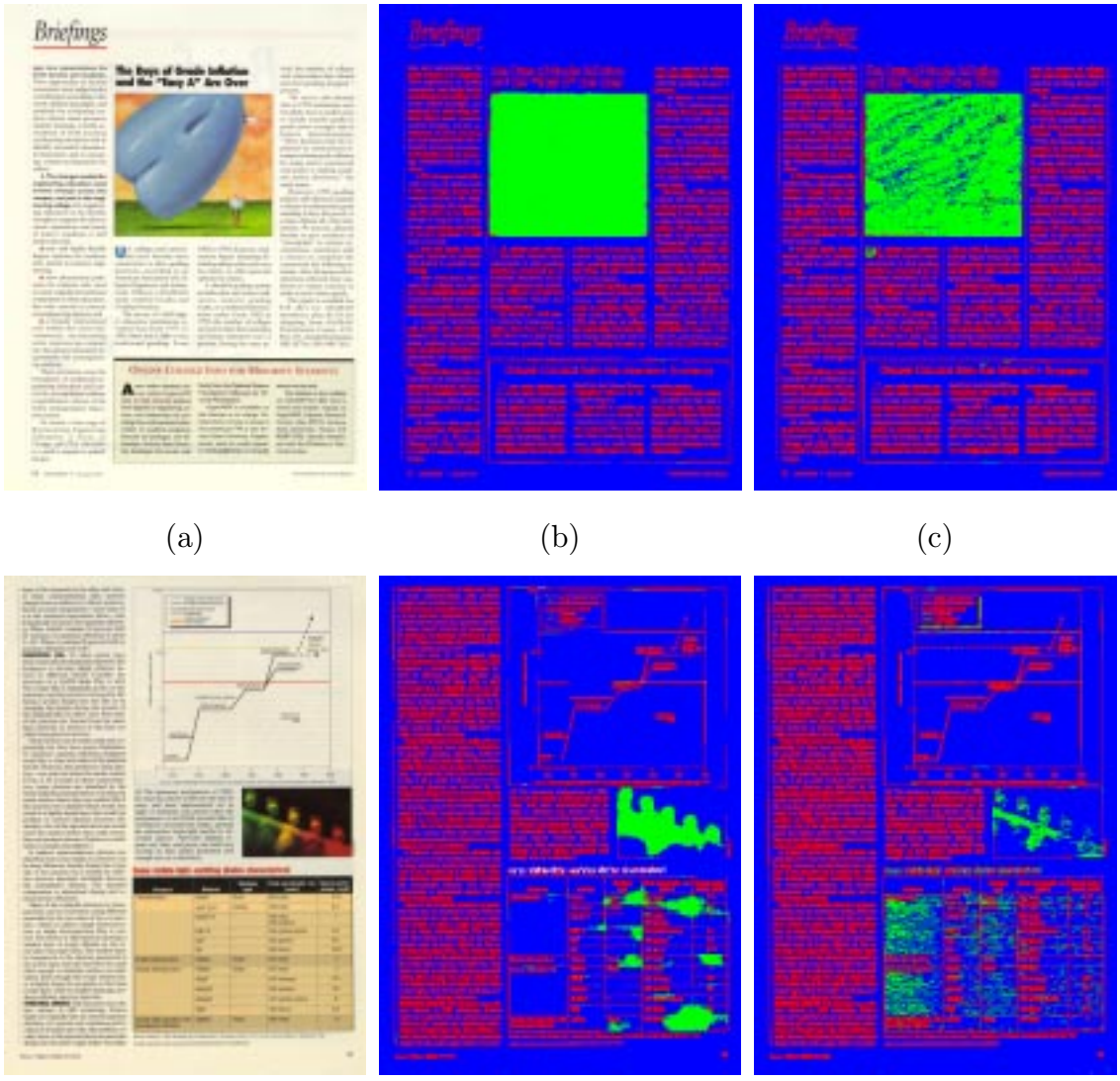
Figure 3.12 compares the RDOS method with DjVu and SPIHT for a small region of scanned text. In general, the quality of text compressed using the RDOS method tends to be better than the other two methods. For example, in Fig. 3.12(c), the text strokes compressed using DjVu look much thicker, such as the "t"s and the "i"s. Fig. 3.13 shows the quality of a scanned picture region compressed using RDOS, DjVu, and SPIHT. The result of the RDOS method generally appears sharper than the results of either of the other two methods.

Fig. 3.14 compares the estimated versus the true bit rates for the three types of coders: One-color, Two-color, and JPEG. The estimates are quite accurate for the One-color class and JPEG class. But for the Two-color class, the estimated rates are substantially higher than the true rates. The reason for this is that we use the JBIG2 compression algorithm for coding binary masks. JBIG2 is a state-of-the-art bi-level image coder, and it exploits the redundancy of a bi-level image at the symbol level. Therefore, it significantly out-performs what can be achieved by the nearest neighbor prediction which is used to estimate the rate of Two-color blocks in RDOS.

3.5 Conclusion

In this chapter, we propose a spatially adaptive compression algorithm for document images which we call the multilayer document compression algorithm. This algorithm first segments a scanned document image into different classes. Then, it compresses each class with an algorithm specifically designed for that class. We also propose a rate-distortion optimized segmentation (RDOS) algorithm for our multilayer document compression algorithm. For each rate-distortion trade-off selected by a user, RDOS chooses the class of each block to optimize the rate-distortion perfor-

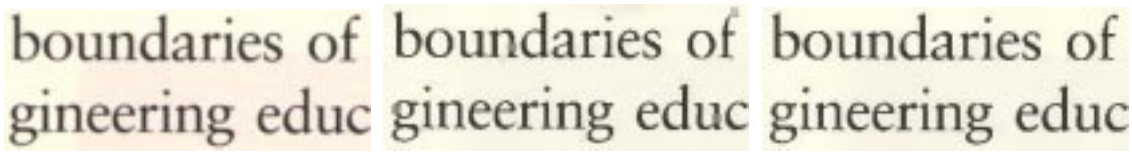
mance over the entire image. Since each block is tested on all coders, RDOS can eliminate severe misclassifications, such as misclassifying a Two-color block as a One-color block. Experimental results show that at similar bit rates, our algorithm can achieve a higher subjective quality than well-known coders such as DjVu, SPIHT and JPEG.



(a) (b) (c)

(d) (e) (f)

Fig. 3.5. Segmentation results of TSMAP and RDOS. (a) Test image I. (b) TSMAP segmentation of test image I, achieved bit rate is 0.138 bpp (173:1 compression). (c) RDOS segmentation of test image I with $\lambda = 0.0021$, achieved bit rate is 0.132 bpp (182:1 compression). (d) Test image II. ©1994 IEEE. Reprinted, with permission, from IEEE Spectrum, page 33, July 1994. (e) TSMAP segmentation of test image II, achieved bit rate is 0.120 bpp (200:1 compression). (f) RDOS segmentation of test image II with $\lambda = 0.0018$, achieved bit rate is 0.114 bpp (210:1 compression). Red, green, blue, white represent Two-color, Picture, One-color, and Other blocks, respectively.



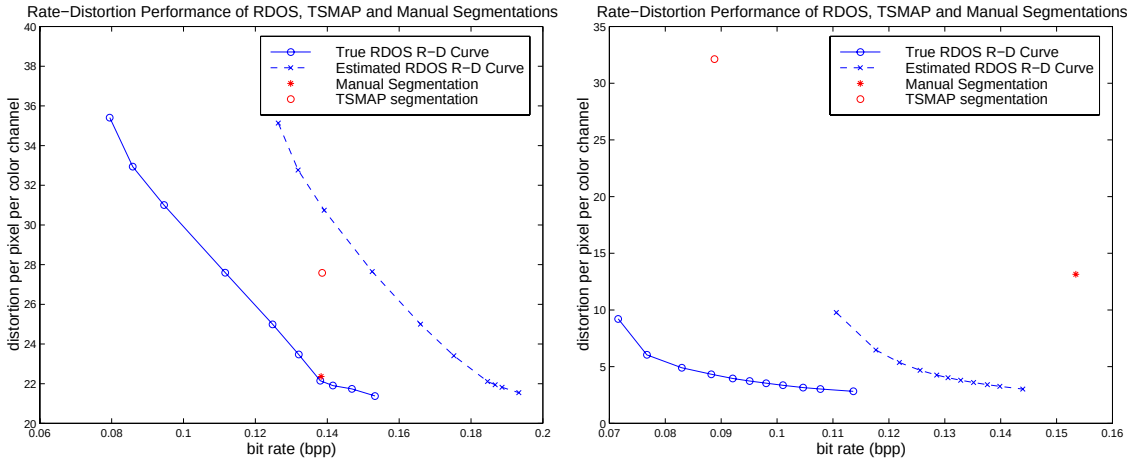
(a) (b) (c)

Fig. 3.6. Comparison between images compressed using the TSMAP segmentation and the RDOS segmentation at similar bit rates. (a) A portion of the original test image I. (b) A portion of the reconstructed image compressed with the TSMAP segmentation at 0.138 bpp (173:1 compression). (c) A portion of the reconstructed image compressed with the RDOS segmentation at 0.132 bpp (182:1 compression), where $\lambda = 0.0021$.



(a) (b) (c)

Fig. 3.7. RDOS segmentations with different λ's. (a) Test image I. (b) RDOS segmentation with $\lambda_1 = 0.0013$, achieved bit rate is 0.095 bpp (253:1 compression). (c) RDOS segmentation with $\lambda_2 = 0.0018$, achieved bit rate is 0.125 bpp (192:1 compression). Red, green, blue, white represent Two-color, Picture, One-color, and Other blocks, respectively.



(a) Test Image I (b) Test Image III

Fig. 3.8. Comparison of rate-distortion performance of the multilayer compression algorithm using RDOS, TSMAP and manual segmentations.



(a) (b) (c)

Fig. 3.9. Test image III and its segmentations. (a) Test image III. (b) RDOS segmentation with $\lambda = 0.0042$, achieved bit rate is 0.101 bpp (237:1 compression). (c) A manual segmentation, achieved bit rate is 0.153 bpp (156:1 compression). Red, green, blue, white represent Two-color, Picture, One-color, and Other blocks, respectively.



(a)

(b)



(c)

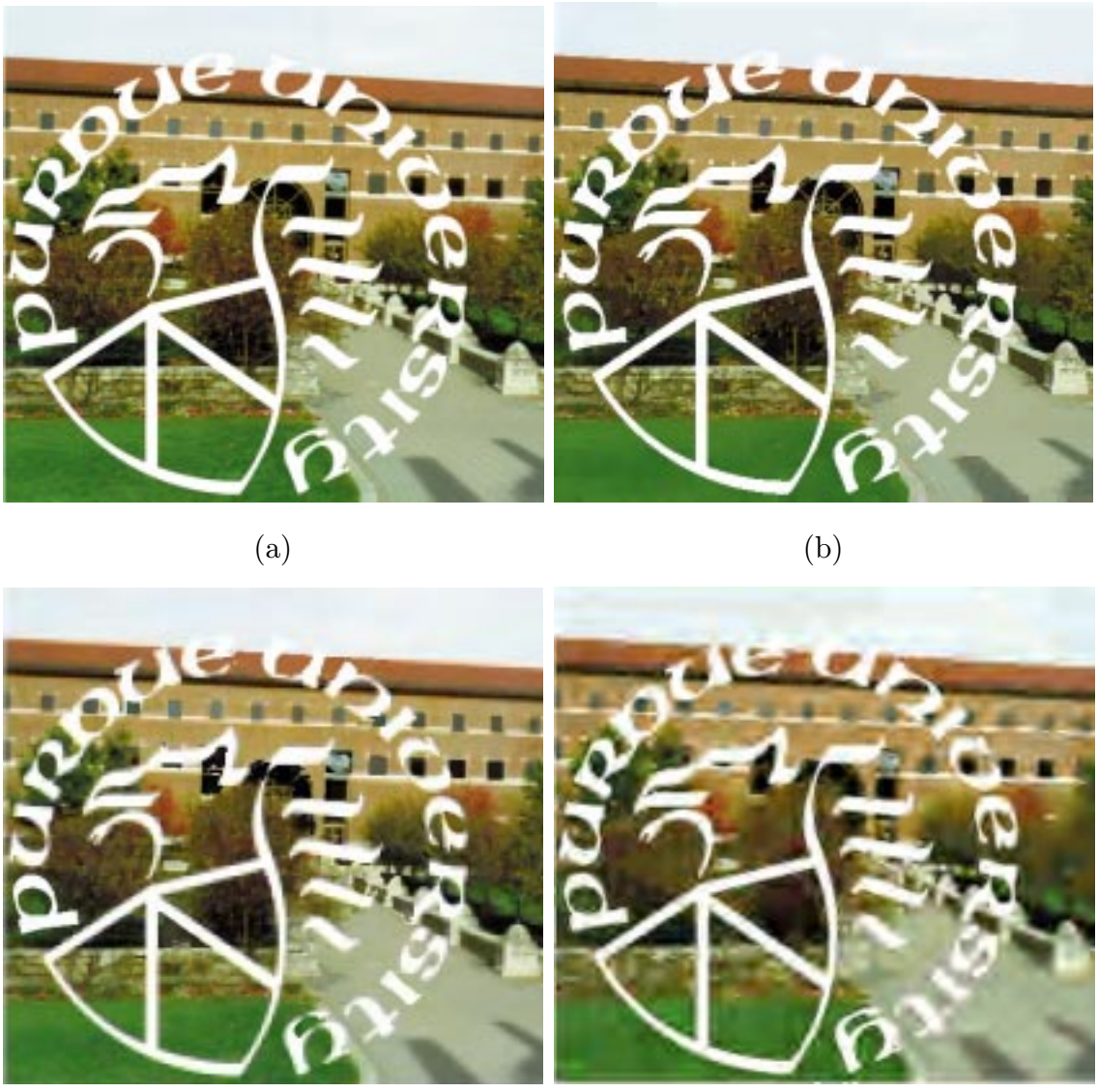


(d)



(e)

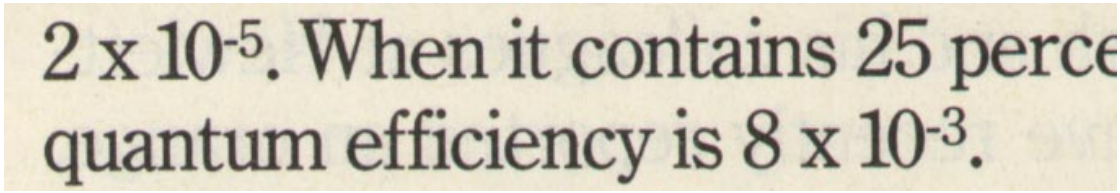
Fig. 3.10. Compression result I. (a) Original image, a portion of test image III. (b) RDOS compressed at 0.101 bpp (237:1 compression), where $\lambda = 0.0042$. (c) DjVu compressed at 0.103 bpp (232:1 compression). (d) SPIHT compressed at 0.103 bpp (233:1 compression). (e) JPEG compressed at 0.184 bpp (131:1 compression).



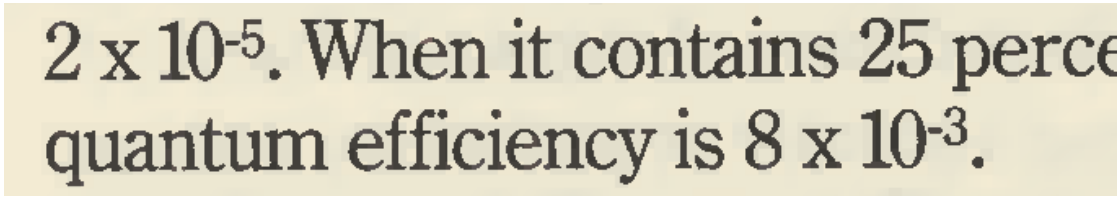
(a) (b)

(c) (d)

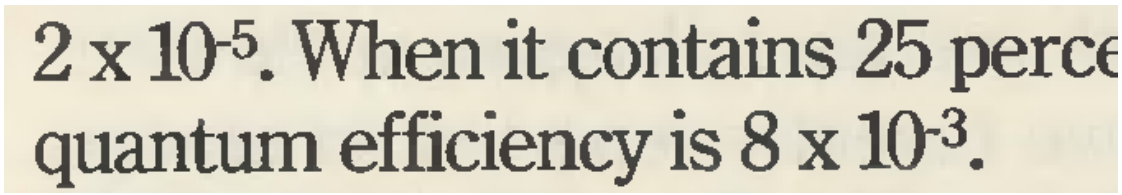
Fig. 3.11. Compression result II. (a) Original image, a portion of test image III. (b) RDOS compressed at 0.101 bpp (237:1 compression), where $\lambda = 0.0042$. (c) DjVu compressed at 0.103 bpp (232:1 compression). (d) SPIHT compressed at 0.103 bpp (233:1 compression).



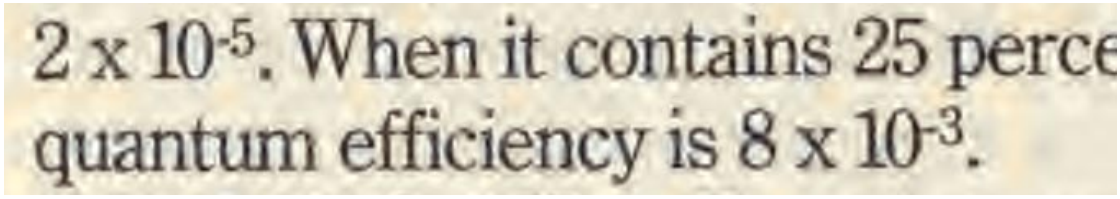
(a)



(b)

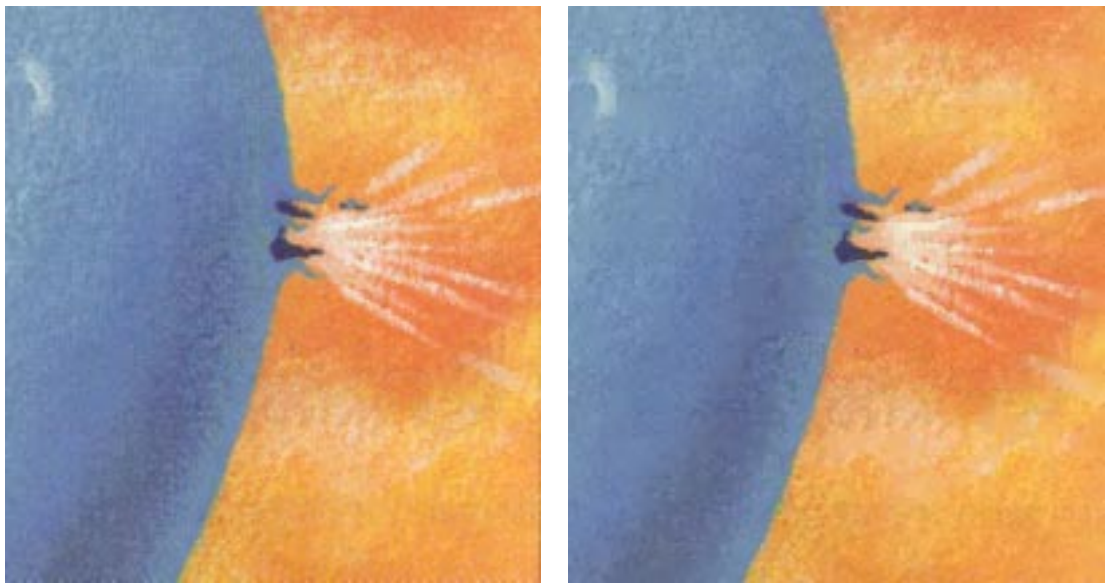


(c)



(d)

Fig. 3.12. Compression result III. (a) Original image, a portion of test image II. (b) RDOS compressed at 0.114 bpp (210:1 compression), where $\lambda = 0.0018$. (c) DjVu compressed at 0.114 bpp (211:1 compression). (d) SPIHT compressed at 0.114 bpp (211:1 compression).



(a)

(b)

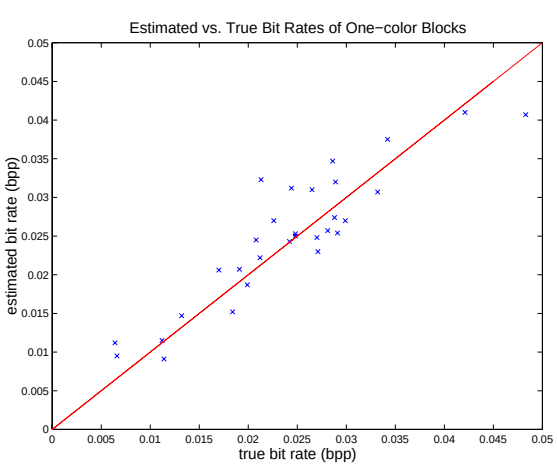


(c)

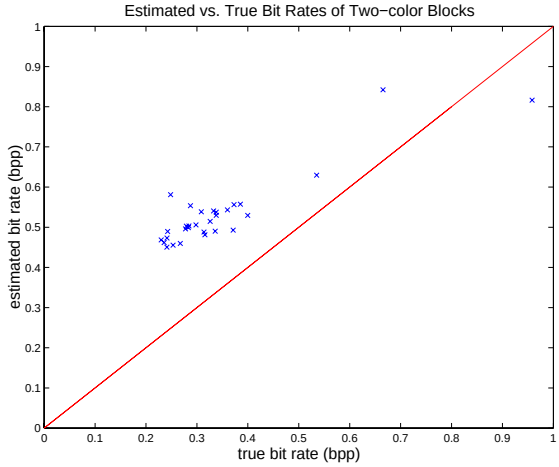


(d)

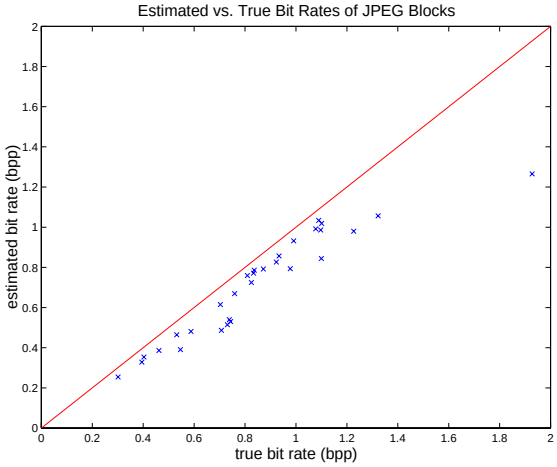
Fig. 3.13. Compression result IV. (a) Original image, a portion of test image I. (b) RDOS compressed at 0.125 bpp (192:1 compression), where $\lambda = 0.0018$. (c) DjVu compressed at 0.132 bpp (182:1 compression). (d) SPIHT compressed at 0.125 bpp (192:1 compression).



(a) One-color Blocks



(b) Two-color Blocks



(c) JPEG Blocks

Fig. 3.14. Estimated vs. true bit rates of coding each class.

LIST OF REFERENCES

[1] K. Y. Wong, R. G. Casey, and F. M. Wahl. Document analysis system. *IBM J. of Res. & Develop.*, 26(6):647–656, November 1982.

[2] D. Wang and S. N. Srihari. Classification of newspaper image blocks using texture analysis. *Comput. Vision Graphics and Image Process.*, 47:327–352, 1989.

[3] P. Chauvet, J. Lopez-Krahe, E. Tafin, and H. Maitre. System for an intelligent office document analysis, recognition and description. *Signal Processing*, 32:161–190, 1993.

[4] R. M. Haralick. Document image understanding: Geometric and logical layout. In *Proc. of IEEE Computer Soc. Conf. on Computer Vision and Pattern Recognition*, volume 8, pages 385–390, Seattle, WA, June 21-23 1994.

[5] K. Murata. Image data compression and expansion apparatus, and image area discrimination processing apparatus therefor. *US Patent 5,535,013*, July 1996.

[6] K. Konstantinides and D. Tretter. A method for variable quantization in JPEG for improved text quality in compound documents. In *Proc. of IEEE Int'l Conf. on Image Proc.*, volume 2, pages 565–568, Chicago, IL, October 4-7 1998.

[7] J. Huang, Y. Wang, and E. K. Wong. Check image compression using a layered coding method. *Journal of Electronic Imaging*, 7(3):426–442, July 1998.

[8] M. Ramos and R. L. de Queiroz. Adaptive rate-distortion-based thresholding: application in JPEG compression of mixed images for printing. In *Proc. of IEEE Int'l Conf. on Image Proc.*, Kobe, Japan, October 25-28 1999.

[9] K. Etemad, D. Doermann, and R. Chellappa. Page segmentation using decision integration and wavelet packets. In *Proc. Int'l Conf. on Pattern Recognition*, volume 2, pages 345–349, Jerusalem, Isr, October 1994.

[10] A. K. Jain and S. Bhattacharjee. Text segmentation using gabor filters for automatic document processing. *Machine Vision and Applications*, 5:196–184, 1992.

[11] A. K. Jain and Y. Zhong. Page segmentation using texture analysis. *Pattern Recognition*, 29(5):743–770, 1996.

[12] C. A. Bouman and M. Shapiro. A multiscale random field model for Bayesian image segmentation. *IEEE Trans. on Image Processing*, 3(2):162–177, March 1994.

[13] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth International Group, Belmont, CA, 1984.

[14] X. Wu and Y. Fang. A segmentation-based predictive multiresolution image coder. *IEEE Trans. on Image Processing*, 4(1):34–47, January 1995.

[15] G. M. Schuster and A. K. Katsaggelos. *Rate-distortion based video compression.* Kluwer Academic Publishers, Boston, 1997.

[16] H. Derin, H. Elliott, R. Cristi, and D. Geman. Bayes smoothing algorithms for segmentation of binary images modeled by Markov random fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-6(6):707–719, November 1984.

[17] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society B*, 48(3):259–302, 1986.

[18] H. Derin and H. Elliott. Modeling and segmentation of noisy and textured images using Gibbs random fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-9(1):39–55, January 1987.

[19] Julian Besag. Efficiency of pseudolikelihood estimation for simple Gaussian fields. *Biometrica*, 64(3):616–618, 1977.

[20] Haluk Derin and Patrick A. Kelly. Discrete-index Markov-type random processes. *Proc. of the IEEE*, 77(10):1485–1510, October 1989.

[21] Jun Zhang, James W. Modestino, and David A. Langan. Maximum-likelihood parameter estimation for unsupervised stochastic model-based image segmentation. *IEEE Trans. on Image Processing*, 3(4):404–420, July 1994.

[22] X. Descombes, R. Morris, J. Zerubia, and M. Berthod. Estimation of Markov random field prior parameters using Markov chain Monte Carlo maximum likelihood. Technical Report 3015, INRIA-Institut National de Recherche en Informatique et en Automatique, October 1996.

[23] Suhail S. Saquib, Charles A. Bouman, and Ken Sauer. ML parameter estimation for Markov random fields with applications to Bayesian tomography. *IEEE Trans. on Image Processing*, 7(7):1029–1044, July 1998.

[24] P. J. Burt, T. Hong, and A. Rosenfeld. Segmentation and estimation of image region properties through cooperative hierarchical computation. *IEEE Trans. on Systems Man and Cybernetics*, SMC-11(12):802–809, December 1981.

[25] I. Ng, J. Kittler, and J. Illingworth. Supervised segmentation using a multiresolution data representation. *Signal Processing*, 31:133–163, March 1993.

[26] C. H. Fosgate, H. Krim, W. W. Irving, W. C. Karl, and A. S. Willsky. Multiscale segmentation and anomaly enhancement of SAR imagery. *IEEE Trans. on Image Processing*, 6(1):7–20, January 1997.

[27] K. Etemad, D. Doermann, and R. Chellappa. Multiscale segmentation of unstructured document pages using soft decision integration. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(1):92–96, January 1997.

[28] M. Unser and M. Eden. Multiresolution feature extraction and selection for texture segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(7):717–728, July 1989.

[29] M. Unser. Texture classification and segmentation using wavelet frames. *IEEE Trans. on Image Processing*, 4(11):1549–1560, November 1995.

[30] E. Salari and Z. Ling. Texture segmentation using hierarchical wavelet decomposition. *Pattern Recognition*, 28(12):1819–1824, December 1995.

[31] Basilis Gidas. A renormalization group approach to image processing problems. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(2):164–180, February 1989.

[32] C. A. Bouman and B. Liu. Multiple resolution segmentation of textured images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(2):99–113, February 1991.

[33] P. Perez and F. Heitz. Multiscale Markov random fields and constrained relaxation in low level image analysis. In *Proc. of IEEE Int'l Conf. on Acoust., Speech and Sig. Proc.*, volume 3, pages 61–64, San Francisco, CA, March 23-26 1992.

[34] C. A. Bouman and M. Shapiro. Multispectral image segmentation using a multiscale image model. In *Proc. of IEEE Int'l Conf. on Acoust., Speech and Sig. Proc.*, volume 3, pages 565–568, San Francisco, California, March 23-26 1992.

[35] J. M. Laferte, F. Heitz, P. Perez, and E. Fabre. Hierarchical statistical models for the fusion of multiresolution image date. In *Proc. Int'l Conf. on Computer Vision*, pages 908–913, Cambridge, MA, June 20-23 1995.

[36] M. S. Crouse, R. D. Nowak, and R. G. Baraniuk. Wavelet-based statistical signal processing using hidden Markov models. *IEEE Trans. on Signal Processing*, 46(4):886–902, April 1998.

[37] Zoltan Kato, Marc Berthod, and Josiane Zerubia. Parallel image classification using multiscale Markov random fields. In *Proc. of IEEE Int'l Conf. on Acoust., Speech and Sig. Proc.*, volume 5, pages 137–140, Minneapolis, MN, April 27-30 1993.

[38] M. L. Comer and E. J. Delp. Segmentation of textured images using a multiresolution Gaussian autoregressive model. *IEEE Trans. on Image Processing*, to appear.

[39] S. B. Gelfand, C. S. Ravishankar, and E. J. Delp. An iterative growing and pruning algorithm for classification tree design. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(2):163–177, February 1984.

[40] H. Cheng, C. A. Bouman, and J. P. Allebach. Multiscale document segmentation. In *Proc. of IS&T's 50th Annual Conf.*, pages 417–425, Cambridge, MA, May 18-23 1997.

[41] H. Cheng and C. A. Bouman. Trainable context model for multiscale segmentation. In *Proc. of IEEE Int'l Conf. on Image Proc.*, volume 1, pages 610–614, Chicago, IL, October 4-7 1998.

[42] J. M. Shaprio. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. on Signal Processing*, 41(12):3445–3462, December 1993.

[43] K. Daoudi, A. B. Frakt, and A. S. Willsky. Multiscale autoregressive models and wavelets. *IEEE Trans. on Information Theory*, to appear.

[44] Murray Aitkin and Donald B. Rubin. Estimation and hypothesis testing in finite mixture models. *Journal of the Royal Statistical Society B*, 47(1):67–75, 1985.

[45] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39(1):1–38, 1977.

[46] O. Ronen, J. R. Rohlicek, and M. Ostendorf. Parameter estimation of dependence tree models using the EM algorithm. *IEEE Signal Processing Letters*, 2(8):157–159, August 1995.

[47] H. Lucke. Bayesian belief networks as a tool for stochatic parsing. *Speech Communication*, 16(1):89–118, January 1995.

[48] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-6:721–741, November 1984.

[49] J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11(2):417–431, September 1983.

[50] S. J. Harrington and R. V. Klassen. Method of encoding an image at full resolution for storing in a reduced image buffer. *US Patent 5,682,249*, October 1997.

[51] R. Buckley, D. Venable, and L. McIntyre. New developments in color facsimile and internet fax. In *Proc. of the Fifth Color Imaging Conference: Color Science, Systems, and Applications*, pages 296–300, Scottsdale, AZ, November 17-20 1997.

[52] 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*, 7(3):410–425, July 1998.

[53] R. L. de Queiroz, R. Buckley, and M. Xu. Mixed raster content (MRC) model for compound image compression. In *Proc. IS&T/SPIE Symp. on Electronic Imaging, Visual Communications and Image Processing*, volume 3653, pages 1106–1117, San Jose, CA, Februray 1999.

[54] H. Cheng and C. A. Bouman. Multiscale document compression algorithm. In *Proc. of IEEE Int'l Conf. on Image Proc.*, Kobe, Japan, October 25-28 1999.

[55] K. Ramchandran and M. Vetterli. Rate-distortion optimal fast thresholding with complete JPEG/MPEG decoder compatibility. *IEEE Trans. on Image Processing*, 3(5):700–704, September 1994.

[56] M. Effros and P. A. Chou. Weighted universal bit allocation: optimal multiple quantization matrix coding. In *Proc. of IEEE Int'l Conf. on Acoust., Speech and Sig. Proc.*, volume 4, pages 2343–2346, Detroit, MI, May 9-12 1995.

[57] A. Ortega and K. Ramchandran. Rate-distortion methods for image and video compression. *IEEE Signal Proc. Magazine*, 15(6):23–50, November 1998.

[58] A. Said and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. on Circ. and Sys. for Video Technology*, 6(3):243–250, June 1996.

[59] M. Nelson and J-L Gailly. *The data compression book*. M & T Books, New York, 1996.

[60] P. G. Howard, F. Kossentini, B. Martins, S. Forchhammer, and W. J. Rucklidge. The emerging JBIG2 standard. *IEEE Trans. on Circ. and Sys. for Video Technology*, 8(7):838–848, November 1998.

[61] Michael Orchard and Charles A. Bouman. Color quantization of images. *IEEE Trans. on Signal Processing*, 39(12):2677–2690, December 1991.

[62] W. B. Pennebaker and J. L. Mitchell. *JPEG: still image date compression standard*. Van Nostrand Reinhold, New York, 1993.

APPENDICES

Appendix A: Computing Log Likelihood Terms

In this appendix, we will derive the recursive formulas for computing $l_s^{(n)}(k)$ which are given in (2.10) and (2.11). For a pixel $s \in S^{(n)}$, we define z_s as the set of pixels which consists of s and its descendents. If we assume the quadtree context model and let

$$l_s^{(n)}(k) = \log p_{\tilde{y}_{z_s}|x_s^{(n)}}(\tilde{y}_{z_s}|k) \ , \tag{A.1}$$

then it is easy to verify that (2.6) holds. When $n \geq 1$, we have

$$\begin{aligned} l_s^{(n)}(k) &= \log p_{\tilde{y}_{z_s}|x_s^{(n)}}(\tilde{y}_{z_s}|k) \\ &= \log p_{\tilde{y}_s^{(n)}|x_s^{(n)}}(\tilde{y}_s^{(n)}|k) \\ &\quad + \sum_{i=1}^{4} \log \left[\sum_{m=0}^{M-1} p_{\tilde{y}_{z_{s_i}}|x_{s_i}^{(n-1)}}(\tilde{y}_{z_{s_i}}|m)\, p_{x_{s_i}^{(n-1)}|x_s^{(n)}}(m|k) \right] \\ &= \log p_{\tilde{y}_s^{(n)}|x_s^{(n)}}(\tilde{y}_s^{(n)}|k) + \sum_{i=1}^{4} \log \left\{ \sum_{m=0}^{M-1} \exp\left[l_{s_i}^{(n-1)}(m) \right] \ \theta_{m,k,n-1} \right\} \end{aligned}$$

where s_i for $i = 1, 2, 3, 4$ are the four children of s. This shows that (2.11) is true.

When $n = 0$, $s \in S^{(0)}$ and $z_s = \{s\}$. Then (A.1) can be rewritten as

$$l_s^{(0)}(k) = \log p_{\tilde{y}_s^{(0)}|x_s^{(0)}}(\tilde{y}_s^{(0}|k) \ .$$

This verifies that (2.10) is true.

Appendix B: Computation of EM Update Using Stochastic Sampling

To compute the EM update using stochastic sampling, the parameters are first initialized to

$$\theta_{i,j,n}^{(0)} = \begin{cases} 0.7 & if \ \ i = j \\ 0.3/(M-1) & if \ \ i \neq j \end{cases}$$

and then we generate samples of $X^{(>0)}$ using a Gibbs sampler [48]. Notice that in the quadtree model, $x_s^{(n)}$ depends only on $x_{\partial s}^{(n+1)}$ and $x_{s_i}^{(n-1)}$, where s_1, s_2, s_3, and s_4 are the four children of s (see Figure 2.8). Therefore, at iteration $j+1$, a sample of $x_s^{(n)}$ can be generated from the conditional probability distribution

$$p_{x_s^{(n)}|x_{\partial s}^{(n+1)},x_{s_i}^{(n-1)}}(k|m,x_{s_i}^{(n-1)}) = \frac{h_s^{(j)}(k,m,n)}{\sum_{l=0}^{M-1} h_s^{(j)}(l,m,n)}$$

where

$$h_s^{(j)}(k,m,n) = \theta_{k,m,n}^{(j)} \prod_{i=1}^{4} \theta_{x_{s_i}^{(n-1)},k,n-1}^{(j)} \ .$$

The Gibbs samples are generated from fine to coarse scales. At each scale, we perform $\lfloor 1.5^n \rfloor$ passes through the samples, so that we only do one pass at the finest scale. Each update of the EM algorithm uses two full fine-to-coarse passes of the Gibbs sampler. After the samples are generated, $\sigma_{k,m,n}^{(j)}$ is estimated by histogramming the $x_s^{(n)}$ results from the two passes of the Gibbs sampler.

$$\sigma_{k,m,n}^{(j)} = \sum_{s \in S^{(n)}} \delta(x_s^{(n)} - k, x_{\partial^{(n)} s} - m)$$

VITA

Hui Cheng was born in Beijing, China in 1969. He received his B.E. in Electrical Engineering, B.S. in Applied Mathematics from Shanghai Jiaotong University in 1991, M.S. in Applied and Computational Mathematics from University of Minnesota in 1995, and Ph.D. in Electrical and Computer Engineering from Purdue University in 1999. From 1991 to 1994, he was with the Institute of Automation, Chinese Academy of Sciences. In 1999, he joined Xerox Corporate Research and Technology.