

# Document Compression Using Rate-Distortion Optimized Segmentation

Hui Cheng and Charles A. Bouman  
School of Electrical and Computer Engineering  
Purdue University  
West Lafayette, IN 47907-1285  
{hui, bouman}@ecn.purdue.edu

**Key words :** document compression    document segmentation  
                 image compression        image segmentation  
                 rate-distortion            multi-resolution  
                 document processing

Corresponding Author : Hui Cheng  
Mailing Address : Xerox Corporation  
                         800 Phillips Road, MS 128-25E  
                         Webster, NY 14580  
Phone : +1 716 422-7704  
Email : Hui.Cheng@crt.xerox.com, hui@ecn.purdue.edu

## Abstract

Effective document compression algorithms require that scanned document images be first segmented into regions such as text, pictures and background. In this paper, we present 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. Two algorithms are investigated for segmenting document images: a direct image segmentation algorithm called the trainable sequential MAP (TSMAP) segmentation algorithm, and a rate-distortion optimized segmentation (RDOS) algorithm. 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.

# 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 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 at very low levels of image 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 crudely classified as block-based approaches and layer-based approaches. Block-based approaches, such as [1, 2, 3, 4], 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 [5, 6, 7, 8] partition a document image into different layers, such as the background layer and the foreground layer. Then, each layer is coded as an image independently 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. However, block-based and layer-based document image representation approaches are closely related. With some overhead, they can be exchanged from one to the other. In addition, they can sometimes be combined to achieve better performance.

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 [1] proposed a method based on the absolute values of DCT coefficients, and Konstantinides and Tretter [3] 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 input document image. The DjVu document compression system [6] uses a multiscale bicolor 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 [4].

In this paper, we present a multilayer document compression algorithm. This algorithm first classifies  $8 \times 8$  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 [9], and a rate-distortion optimized segmentation (RDOS) algorithm developed for document compression [10].

The TSMAP algorithm 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 [11] proposed a rate-distortion optimal technique to drop quantized DCT coefficients of a JPEG or an MPEG coder. Effros and Chou [12] introduced a two-stage bit allocation algorithm for a simple DCT-based source coder.<sup>1</sup> 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 [13] 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 [14].

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 than switching among sets of parameters for a fixed vector quantizer (VQ), DCT, or Karhunen-Loeve

---

<sup>1</sup>The DCT-based coder used in [12] differs from JPEG because the DC component is not differentially encoded, and no zigzag run-length encoding of the AC components is used.

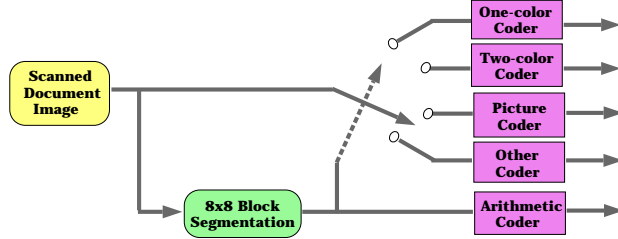


Figure 1: General structure of the multilayer document compression algorithm.

(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 Klassen [2]. By exploiting the bi-level nature of text, this coder gives performance which is far superior to what can be achieved with transform coders. Another distinction of our method is that different coders use somewhat different distortion measures. This is motivated by the fact that perceived quality for text, graphics and pictures can be quite different. A class-dependent distortion measure is also found valuable in [4]. Our approach is similar in concept to the one proposed by Reusens et al. for MPEG-4 video coding [15], where five compression models are used for videoconference applications: motion compensation model, background model, bi-color text and graphics model, DCT model and fractal model. However, they use a square-error distortion measure.

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 [16] and JPEG.

## 2 Multilayer Compression Algorithm

The multilayer compression algorithm shown in Fig. 1 classifies each  $8 \times 8$  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

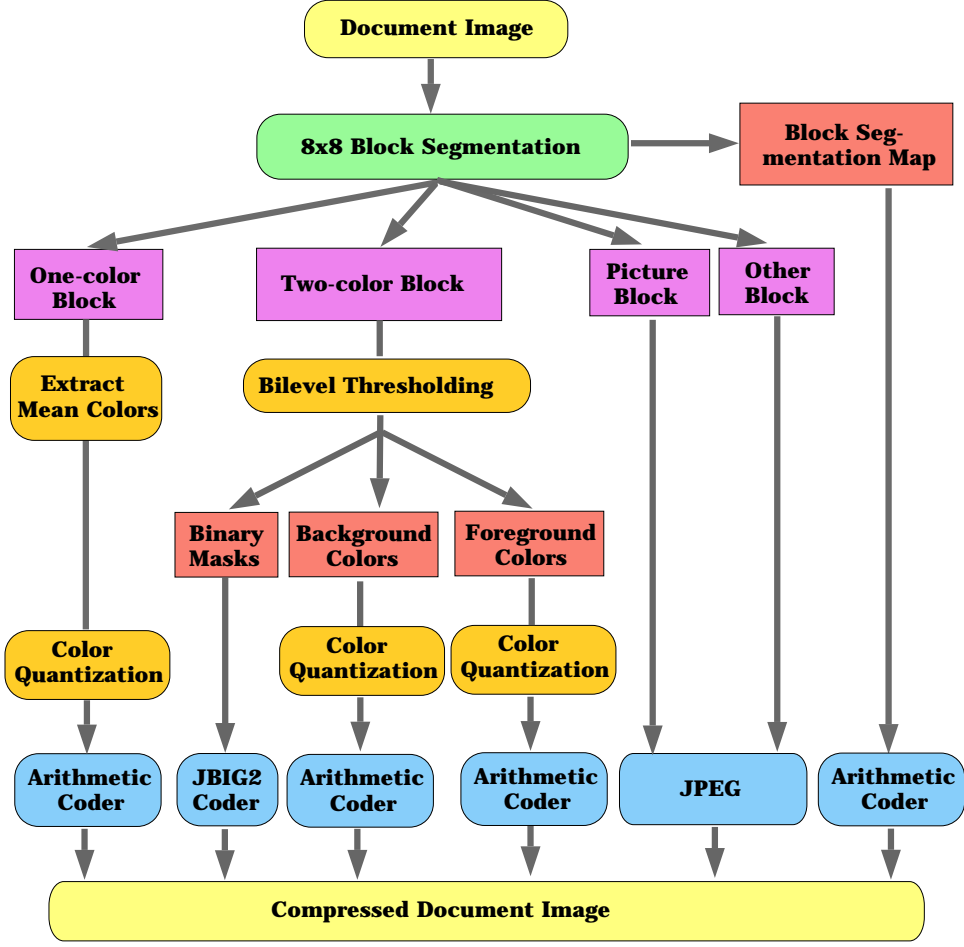


Figure 2: Flow diagram of the multilayer document compression algorithm.

class labels of all blocks are compressed and sent as side information.

The flow diagram of our compression algorithm is shown in Fig. 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 graphics, and they need to be coded with high spatial resolution. Therefore, for each Two-color block, a bilevel 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 the arithmetic coders. 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 paper, we use  $y$  to denote the original image and  $x$  to denote its  $8 \times 8$  block segmentation. Also,  $y_i$  denotes the  $i$ -th  $8 \times 8$  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.

## 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 an arithmetic coder based on a third order Markov model [17]. 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.

## 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 bilevel 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,  $8 \times 8$  blocks are used. But sometimes an

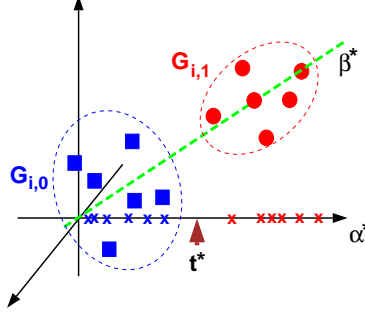


Figure 3: Minimal MSE thresholding. We use  $\alpha^*$  to denote the color axis with the largest variance, and  $\beta^*$  to denote the principle axis.  $t^*$  is the optimal threshold on  $\alpha^*$ , and x's are the samples projected on  $\alpha^*$ .

$8 \times 8$  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  $16 \times 16$  block centered at the  $8 \times 8$  block will be used instead.

The minimal MSE thresholding algorithm is illustrated in Fig. 3. For a Two-color block  $y_i$ , we first project all colors of  $y_i$  onto the color axis  $\alpha^*$  which has the largest variance among three color axes. The thresholding is done only on  $\alpha^*$ . 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 bilevel thresholding. For a threshold  $t$  on  $\alpha^*$ ,  $t$  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  $8 \times 8$ , we will enlarge the block to  $16 \times 16$  symmetrically along all directions, and use the same bilevel thresholding algorithm to extract two colors and a  $16 \times 16$  mask. Then, the two colors extracted from the  $16 \times 16$  block are used as  $c_{i,0}$  and  $c_{i,1}$ , and middle portion of the  $16 \times 16$  mask is used as  $b_{i,m,n}$ . If  $\tilde{G}_{i,j}$  is empty, and the current block is a  $16 \times 16$  block,  $c_{i,j}$  will be used as it is without refinement.

After bilevel 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 an arithmetic coder based on a third order Markov model. The color indices of background colors are compressed similarly.

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 [18].

### 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  $2 \times 2$  subsampling of the two chrominance channels to reduce the overall bit rate. This means that each  $8 \times 8$  JPEG chrominance block will correspond to 4 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 or the RDOS methods of segmentation, and will be discussed in detail in sections 3.1 and 3.2.

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 than 4 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 at the end of packed luminance blocks, so that we can use the standard JPEG library routines provided by the Independent JPEG Group.

## 2.4 Additional Issues

The block segmentation  $x$  for the luminance blocks is entropy coded using an arithmetic coder based on a third order Markov model. 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 using the arithmetic coder.

As stated above, the Two-color blocks and One-color blocks use color quantization as a pre-processing 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 [19] 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 than a threshold of 10 for the One-color blocks and 30 for the Two-color blocks.

### 3 Segmentation Algorithms

In order to better understand the role of segmentation in document compression, we will compare two different types of segmentation algorithms: the trainable sequential MAP (TSMAP) algorithm of [9], and the rate distortion optimized segmentation (RDOS) described in the following section. The TSMAP is representative of a broad class of direct segmentation algorithms that segment the document based solely on the document image. In contrast, the RDOS method works in a closed loop fashion by applying each coding method to each region of the document and then selecting the method that yields that best rate-distortion trade-off.

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. The following sections give details of the TSMAP and RDOS methods.

#### 3.1 TSMAP Segmentation Algorithm

The first segmentation algorithm used for the multilayer document compression is the TSMAP segmentation algorithm [9]. The TSMAP algorithm is based on the multiscale Bayesian approach proposed by Bouman and Shapiro [20]. Fig. 4 shows the TSMAP segmentation model. It has a novel multiscale context model which can capture complex aspects of both local and global

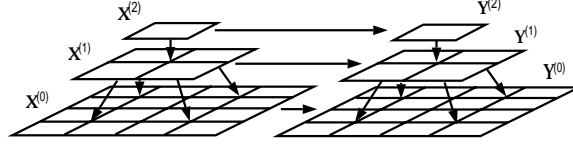


Figure 4: The TSMAP segmentation model. The left pyramid models the contextual behavior, while the right pyramid models the data features extracted using a Haar wavelet transform.  $X^{(n)}$  are class labels at scale  $n$ , and  $Y^{(n)}$  are image feature vectors extracted at scale  $n$ .

contextual behavior. In addition, our multiscale image model uses local texture features extracted via a wavelet decomposition, and the textural information at various scales is captured.

The parameters which describe the characteristics of typical images are extracted from a database of training images which are produced by scanning typical images 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 the multilayer document compression algorithm, 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 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 the background color  $c_{i,0}$  and the foreground color  $c_{i,1}$ . 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  $8 \times 8$  block segmentation 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  $8 \times 8$  block segmentation.

### 3.2 Rate-Distortion Optimized Segmentation

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  $D(y|x)$  be the total distortion resulting from coding  $y$  with segmentation  $x$ . Then, the rate-distortion optimized segmentation,  $x^*$ , is

$$x^* = \arg \min_{x \in \mathcal{N}^L} \{R(y|x) + R(x) + \lambda D(y|x)\}, \quad (1)$$

where  $\lambda$  is a non-negative real number which controls the trade-off between bit rate and distortion. In our approach, we assume that  $\lambda$  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, (1) can be rewritten as

$$x^* = \arg \min_{\{x_0, x_1, \dots, x_{L-1}\} \in \mathcal{N}^L} \sum_{i=0}^{L-1} \{R_i(x_i|x_{i-1}) + R_x(x_i|x_{i-1}) + \lambda D_i(x_i)\}, \quad (2)$$

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 using each coder, (2) can be solved by a dynamic programming technique similar to that used in [13].

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 expressions 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 to code, typically less than 0.01 bits per color pixel.

### 3.2.1 One-color Blocks

Recall from section 2.1 that each One-color block is represented by an indexed color. Color indices of all One-color blocks are entropy coded with an arithmetic coder based on a third order Markov model. 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  $\mu_i$  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  $x_i = \text{One}$ , 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  $y_i$ ,  $0 \leq m, n \leq 7$ , and  $\|a\| = \sqrt{a^t a}$ .

### 3.2.2 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 neighbors, denoted as  $V_{i,m,n} = [b_{i,m-1,n-1}, b_{i,m-1,n}, b_{i,m-1,n+1}, b_{i,m,n-1}]^t$ . 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 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  $8 \times 8$  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 boundary points. On the other hand, if a block is not a Two-color block, a third color

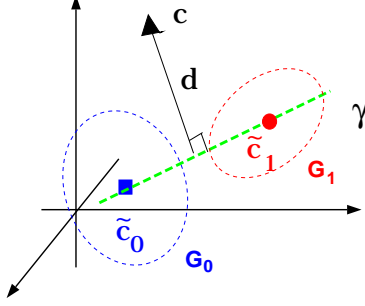


Figure 5: Two-color distortion measure.  $\tilde{c}_0$  and  $\tilde{c}_1$  are indexed mean colors of group  $G_0$  and  $G_1$ , respectively.  $\gamma$  is the line determined by  $\tilde{c}_0$  and  $\tilde{c}_1$ . The distance between a color  $c$  and  $\gamma$  is  $d$ . When  $c$  is a combination of  $\tilde{c}_0$  and  $\tilde{c}_1$ ,  $d = 0$ .

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. 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  $x_i = Two$ ,

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

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. 5, 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.



### 3.2.3 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 JPEG Huffman tables for luminance,  $R_i^l(x_i|x_{i-1})$  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 [21]. 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  $2 \times 2$ , we approximate the number of bits for coding the chrominance components of an  $8 \times 8$  block  $i$ ,  $R_i^c(x_i|x_{i-1})$ , as follows. Let  $j$  be the index of the  $16 \times 16$  block which contains block  $i$ . Also, let  $\beta_{j,k}^d(z_j)$  be the quantized DC coefficient of the  $k$ -th chrominance component using the chrominance quantization table of class  $z_j$ , and  $\beta_{j,k}^a(z_j)$  be the vector of the quantized AC coefficients. Then, we assume that

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

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  $8 \times 8$  blocks of the input document image, and assume that the classes of the chrominance blocks  $j, j-1$  are  $x_i$  and  $x_{i-1}$ , respectively.

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  $e_i^l(x_i)$  be the quantization error of luminance DCT coefficients of block  $i$  using the luminance quantization table of  $x_i$ , and  $e_{j,k}^c(z_j)$  be the quantization error of DCT coefficients of the  $k$ -th chrominance component of the  $16 \times 16$  block containing block  $i$  using the chrominance quantization table of  $z_j$ . Then, the distortion is approximately given by

$$D_i(x_i) = \|e_i^l(x_i)\|^2 + \sum_{k=0}^1 \|e_{j,k}^c(x_i)\|^2.$$

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

In RDOS, the chrominance segmentation is not computed from the  $8 \times 8$  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  $16 \times 16$  block in raster order. We first compute a  $16 \times 16$  block segmentation  $z = \{z_0, z_1, \dots, z_{L/4-1}\}$  which is rate-distortion optimized with 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  $z_{j-1}$ ,

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

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

$$\tilde{D}_j(z_j) = \sum_{k=0}^1 \|e_{j,k}^c(z_j)\|^2$$

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

image	segmentation algorithm	bit rate (bbp)	compression ratio	RDOS distortion per pixel per color	$\lambda$
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 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.

## 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 (bbp) 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. 10 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 [9]. These parameters were extracted from a separate set of 20 manually segmented grayscale images scanned at 100 dpi.

Fig. 6(a) and (d) show the original test image I and test image II<sup>2</sup>. Their TSMAP segmentations are shown in Fig. 6(b) and (e). Fig. 6(c) is the RDOS segmentation of test image I with  $\lambda = 0.0021$ , and Fig. 6(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 1.

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

---

<sup>2</sup>©1994 IEEE. Reprinted, with permission, from IEEE Spectrum, page 33, July 1994.

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 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  $\lambda = 0.0018$ .

TSMAP segmentations appear better than RDOS segmentations with 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. 6 (e), several line segments in the graphics are misclassified as One-color blocks.

In Fig. 7, we compare the quality of reconstructed images compressed using both the TSMAP segmentation and the RDOS segmentation at similar bit rates. Figures 7(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. 7(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 2, we list the average bit rate and standard deviation of each 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  $8 \times 8$  block segmentation and the chrominance segmentation.

Figure 8 shows the RDOS segmentations of test image I using different  $\lambda$ 's, where  $\lambda_1 = 0.013$  and  $\lambda_2 = 0.018$ . It can be seen that for smaller  $\lambda$ , less weight is put on the distortion, and more blocks are segmented as One-color blocks. When  $\lambda$  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  $\lambda$  changes within a reasonable range.

In Fig. 9, we compare the rate-distortion performance achieved by the multilayer compression algorithm using RDOS, TSMAP and manual segmentations. Figure 9(a) is computed from test image I shown in Fig. 6(a), and Fig. 9(b) is computed from test image III, the synthetic image shown in Fig. 10(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 3.2. The solid lines in Fig. 9 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. 10(c) has rate-distortion performance which is inferior to which is achieved by the RDOS segmentation. Both the RDOS and the manual segmentations result in superior rate-distortion performance to the TSMAP segmentations.

Figures 11–14 compare, at similar bit rates, the quality of the reconstructed images compressed using RDOS segmentation with those compressed using three well-known coders: DjVu, SPIHT [16], 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 bilevel 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 11 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. 12(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 13 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. 13(c), the text strokes compressed using DjVu look much thicker, such as the “t”s and the “i”s. This artifact may be caused by the intentional thickening of letters to increase readability in DjVu. This thickening can be turned off by the option “-t” in DjVu. However, all algorithms are run with their default settings. Fig. 14 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. 15 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 bilevel image coder, and it exploits the redundancy of a bilevel 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.

## 5 Conclusion

In this paper, 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 performance 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.

## 6 Acknowledgment

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

## References

- [1] K. Murata, “Image data compression and expansion apparatus, and image area discrimination processing apparatus therefor,” *US Patent 5,535,013*, July 1996.
- [2] 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.

- [3] K. Konstantinides and D. Tretter, "A method for variable quantization in JPEG for improved text quality in compound documents," *Proc. of IEEE Int'l Conf. on Image Proc.*, vol. 2, October 4-7 1998, Chicago, IL, pp. 565–568.
- [4] M. Ramos and R. L. de Queiroz, "Adaptive rate-distortion-based thresholding: application in JPEG compression of mixed images for printing," *Proc. of IEEE Int'l Conf. on Image Proc.*, October 25-28 1999, Kobe, Japan.
- [5] R. Buckley, D. Venable, and L. McIntyre, "New developments in color facsimile and internet fax," *Proc. of the Fifth Color Imaging Conference: Color Science, Systems, and Applications*, November 17-20 1997, Scottsdale, AZ, pp. 296–300.
- [6] L. Bottou, P. Haffner, P. G. Howard, P. Simard, Y. Bengio, and Y. LeCun, "High quality document image compression with 'DjVu'," *Journal of Electronic Imaging*, vol. 7, no. 3, pp. 410–425, July 1998.
- [7] J. Huang, Y. Wang, and E. K. Wong, "Check image compression using a layered coding method," *Journal of Electronic Imaging*, vol. 7, no. 3, pp. 426–442, July 1998.
- [8] R. L. de Queiroz, R. Buckley, and M. Xu, "Mixed raster content (MRC) model for compound image compression," *Proc. IS&T/SPIE Symp. on Electronic Imaging, Visual Communications and Image Processing*, vol. 3653, February 1999, San Jose, CA, pp. 1106–1117.
- [9] H. Cheng and C. A. Bouman, "Trainable context model for multiscale segmentation," *Proc. of IEEE Int'l Conf. on Image Proc.*, vol. 1, October 4-7 1998, Chicago, IL, pp. 610–614.
- [10] H. Cheng and C. A. Bouman, "Multiscale document compression algorithm," *Proc. of IEEE Int'l Conf. on Image Proc.*, October 25-28 1999, Kobe, Japan.
- [11] K. Ramchandran and M. Vetterli, "Rate-distortion optimal fast thresholding with complete JPEG/MPEG decoder compatibility," *IEEE Trans. on Image Processing*, vol. 3, no. 5, pp. 700–704, September 1994.



- [12] M. Effros and P. A. Chou, "Weighted universal bit allocation: optimal multiple quantization matrix coding," *Proc. of IEEE Int'l Conf. on Acoust., Speech and Sig. Proc.*, vol. 4, May 9-12 1995, Detroit, MI, pp. 2343-2346.
- [13] G. M. Schuster and A. K. Katsaggelos, *Rate-distortion based video compression*. Boston: Kluwer Academic Publishers, 1997.
- [14] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Proc. Magazine*, vol. 15, no. 6, pp. 23-50, November 1998.
- [15] E. Reusens, T. Ebrahimi, C. L. Buhan, R. Castagno, V. Vaerman, L. Piron, C. de Sola Fabregas, S. Bhattacharjee, F. Bossen, and M. Kunt, "Dynamic approach to visual data compression," *IEEE Trans. on Circ. and Sys. for Video Technology*, vol. 7, no. 1, pp. 197-211, February 1997.
- [16] 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*, vol. 6, no. 3, pp. 243-250, June 1996.
- [17] M. Nelson and J.-L. Gailly, *The data compression book*. New York: M & T Books, 1996.
- [18] 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*, vol. 8, no. 7, pp. 838-848, November 1998.
- [19] M. Orchard and C. A. Bouman, "Color quantization of images," *IEEE Trans. on Signal Processing*, vol. 39, no. 12, pp. 2677-2690, December 1991.
- [20] C. A. Bouman and M. Shapiro, "A multiscale random field model for Bayesian image segmentation," *IEEE Trans. on Image Processing*, vol. 3, no. 2, pp. 162-177, March 1994.
- [21] W. B. Pennebaker and J. L. Mitchell, *JPEG: still image data compression standard*. New York: Van Nostrand Reinhold, 1993.

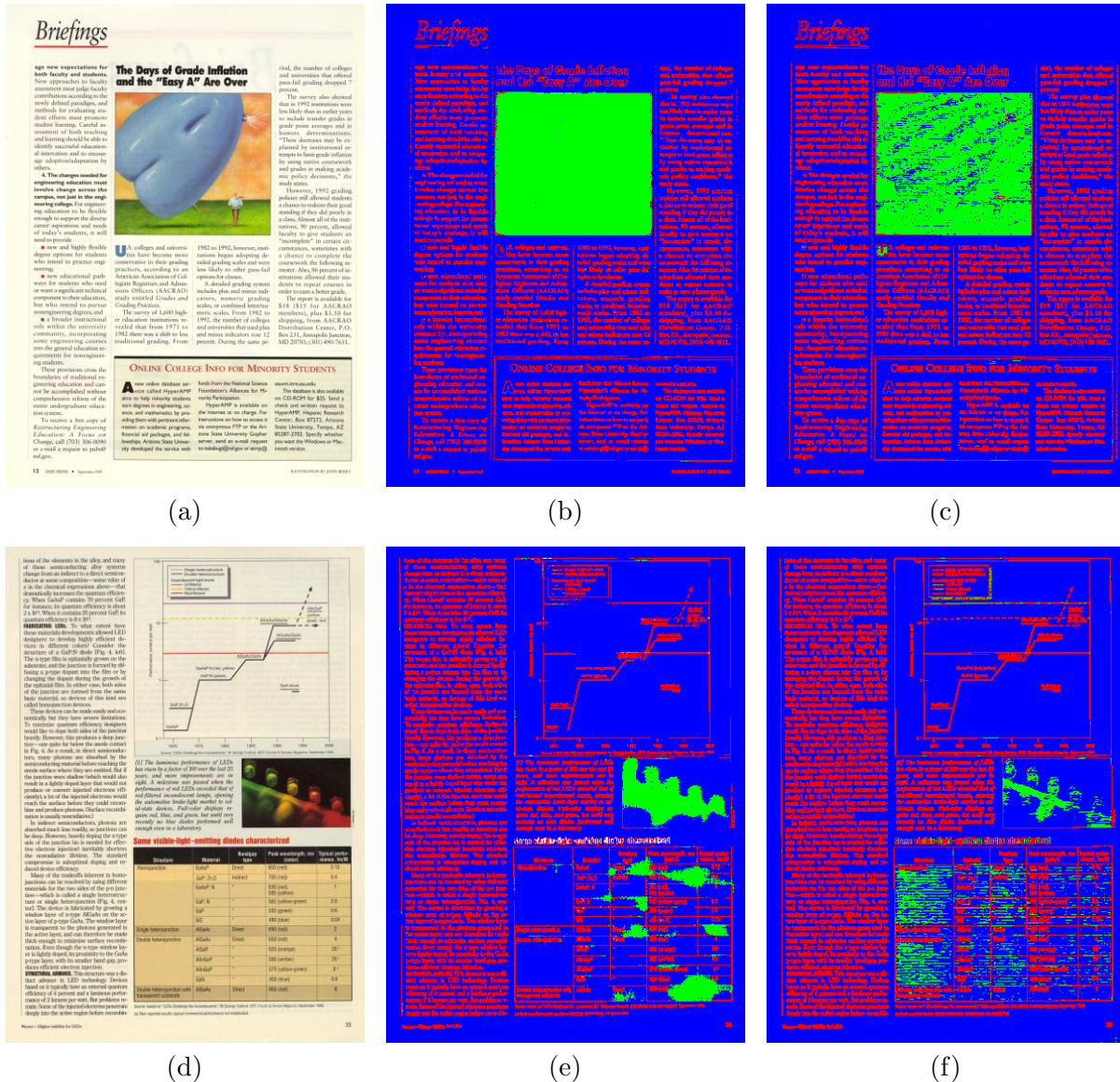


Figure 6: Segmentation results of TSMAP and RDOS. (a) Test image I. (b) TSMAP segmentation of test image I, achieved bit rate is 0.138 bpp (173:1 compression). (c) RDOS segmentation of test image I with  $\lambda = 0.0021$ , achieved bit rate is 0.132 bpp (182:1 compression). (d) Test image II. ©1994 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.

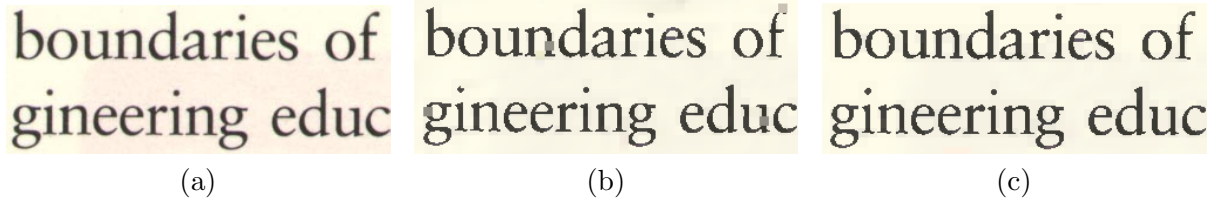


Figure 7: Comparison between images compressed using the TSMAP segmentation and the RDOS segmentation at similar bit rates. (a) A portion of the original test image I. (c) A portion of the reconstructed image compressed with the TSMAP segmentation at 0.138 bpp (173:1 compression). (b) A portion of the reconstructed image compressed with the RDOS segmentation at 0.132 bpp (182:1 compression), where  $\lambda = 0.0021$ .

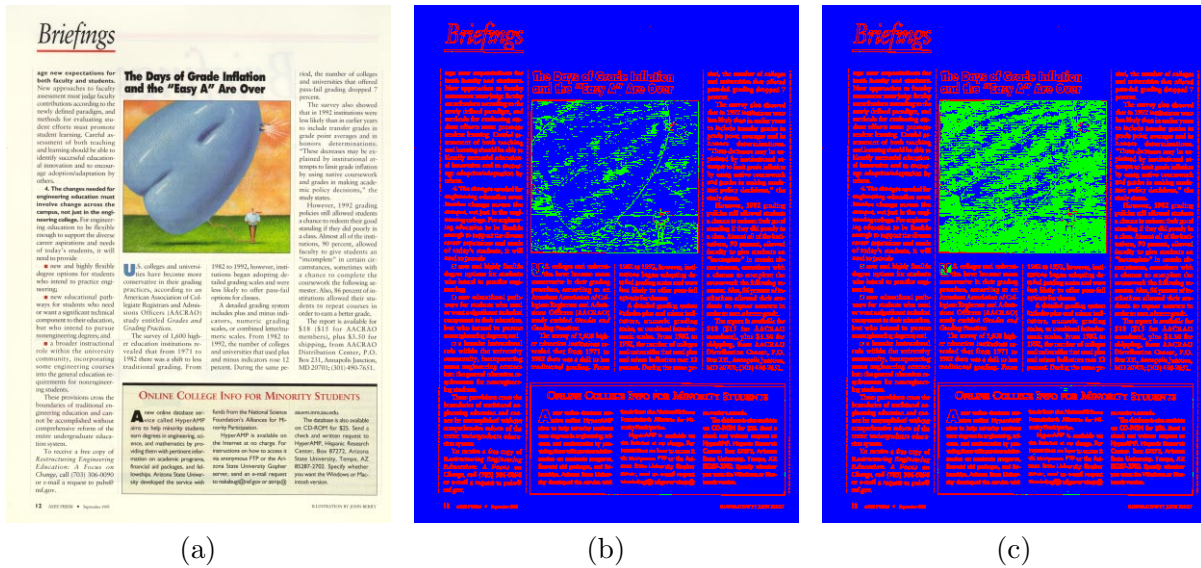


Figure 8: RDOS segmentations with different  $\lambda$ 's. (a) Test image I. (b) RDOS segmentation with  $\lambda_1 = 0.0013$ , achieved bit rate is 0.095 bpp (253:1 compression). (c) RDOS segmentation with  $\lambda_2 = 0.0018$ , achieved bit rate is 0.125 bpp (192:1 compression). Red, green, blue, white represent Two-color, Picture, One-color, and Other blocks, respectively.

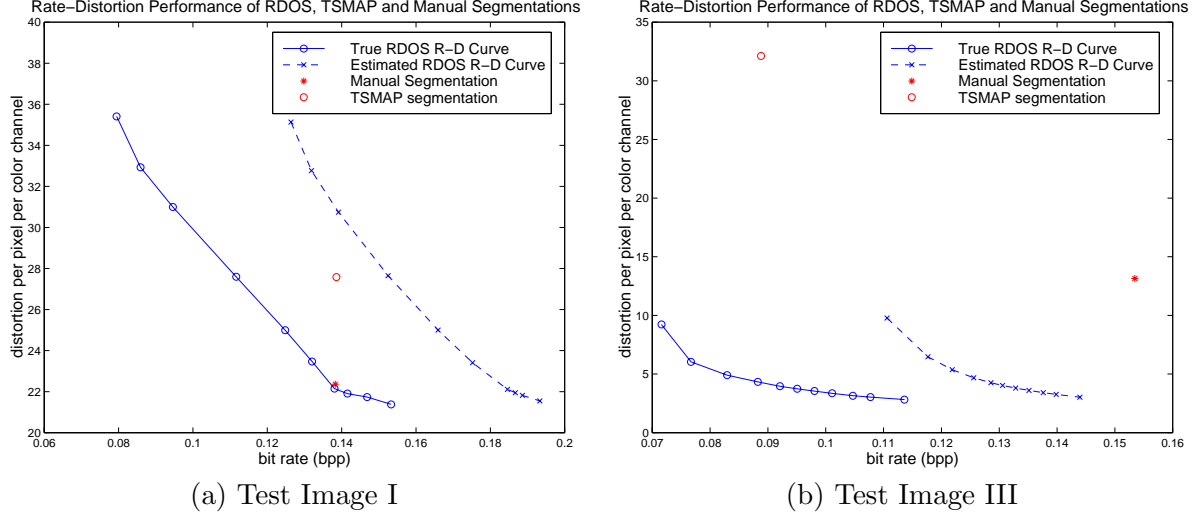


Figure 9: Comparison of rate-distortion performance of the multilayer document compression algorithm using RDOS, TSMAP and manual segmentations.

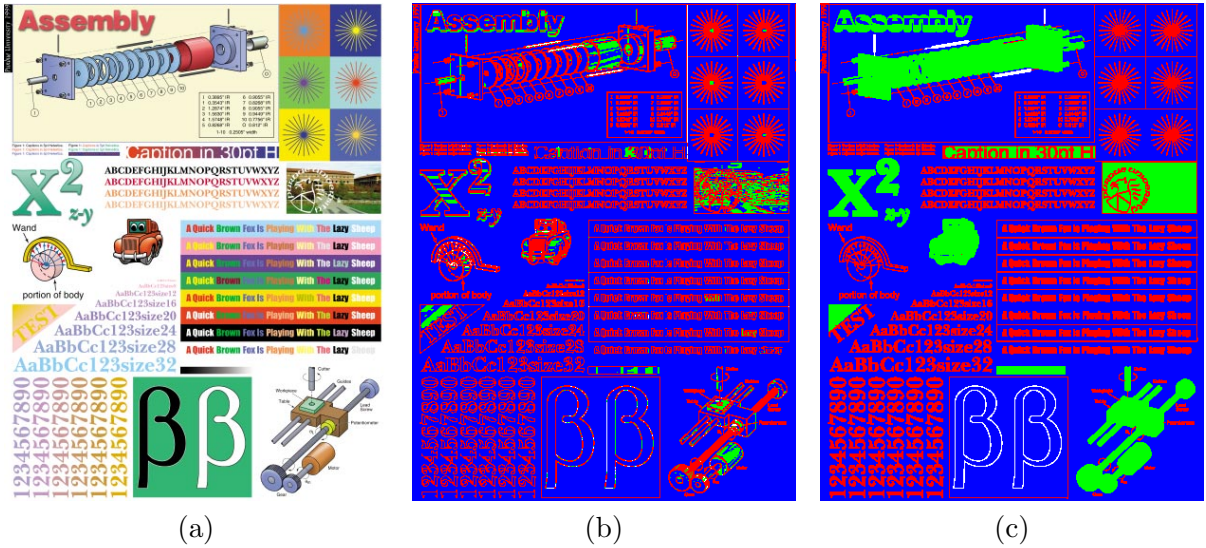


Figure 10: Test image III and its segmentations. (a) Test image III. (b) RDOS segmentation with  $\lambda = 0.0042$ , achieved bit rate is 0.101 bpp (237:1 compression). (c) A manual segmentation, achieved bit rate is 0.153 bpp (156:1 compression). Red, green, blue, white represent Two-color, Picture, One-color, and Other blocks, respectively.



(a)



(b)



(c)



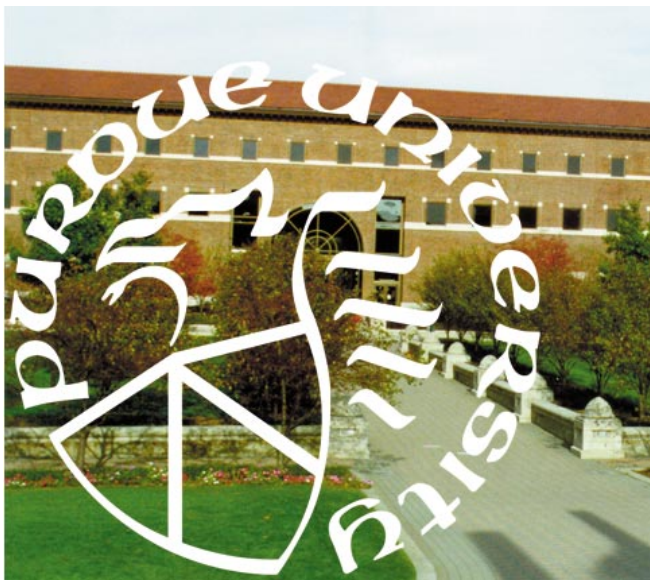
(d)



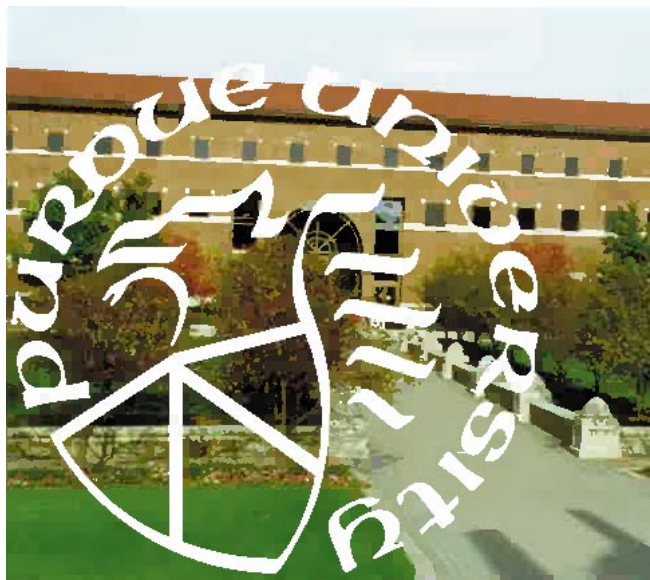
(e)

Figure 11: Compression result I. (a) A portion of the original 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).

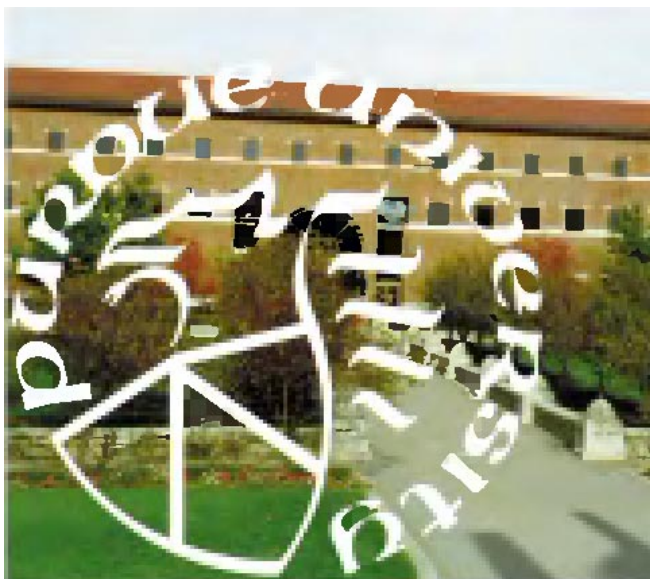




(a)



(b)

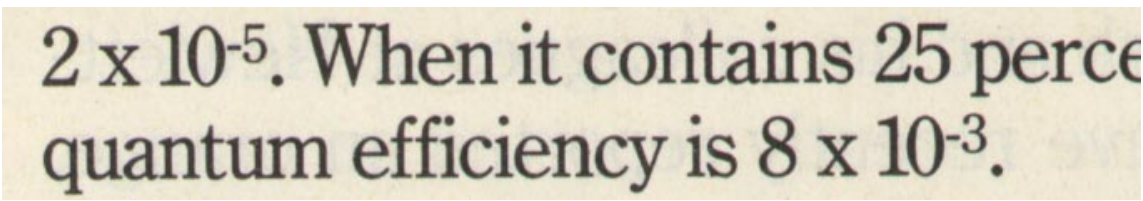


(c)



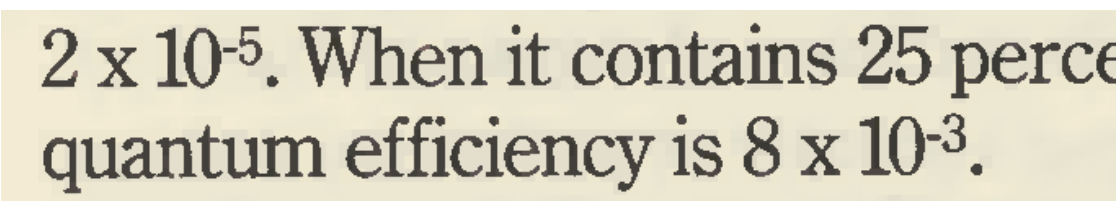
(d)

Figure 12: Compression result II. (a) A portion of the original 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).



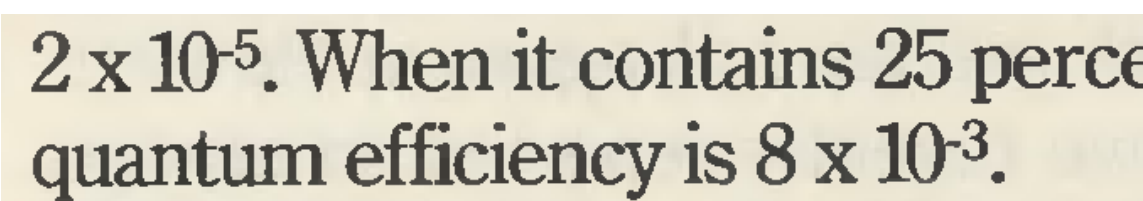
$2 \times 10^{-5}$ . When it contains 25 perce  
quantum efficiency is  $8 \times 10^{-3}$ .

(a)



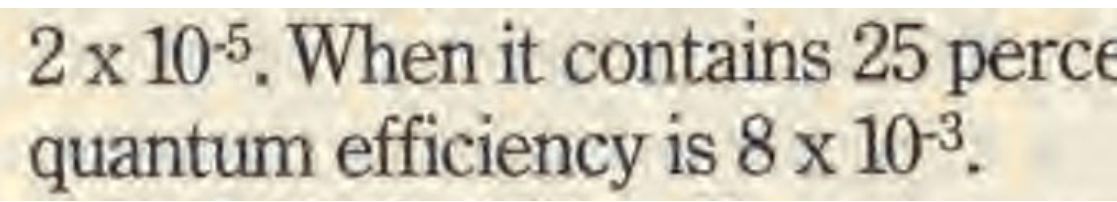
$2 \times 10^{-5}$ . When it contains 25 perce  
quantum efficiency is  $8 \times 10^{-3}$ .

(b)



$2 \times 10^{-5}$ . When it contains 25 perce  
quantum efficiency is  $8 \times 10^{-3}$ .

(c)



$2 \times 10^{-5}$ . When it contains 25 perce  
quantum efficiency is  $8 \times 10^{-3}$ .

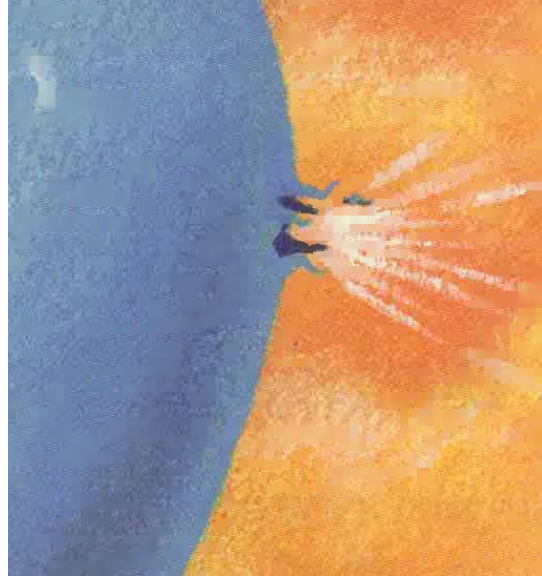
(d)

Figure 13: Compression result III. (a) A portion of the original 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)



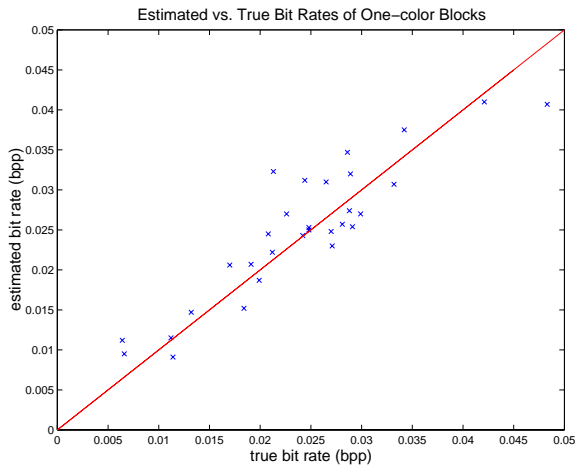
(c)



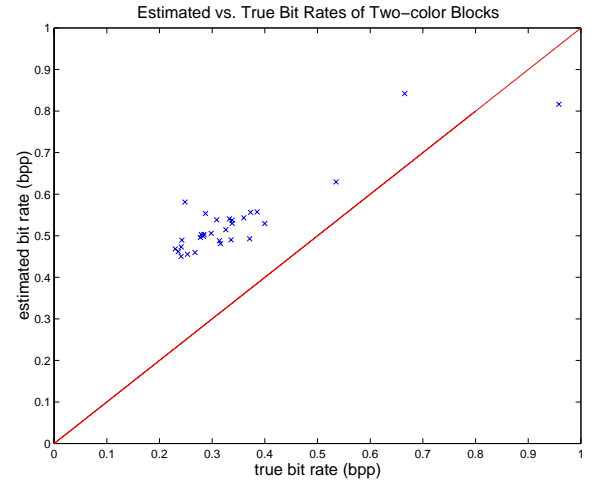
(d)

Figure 14: Compression result IV. (a) A portion of the original 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).

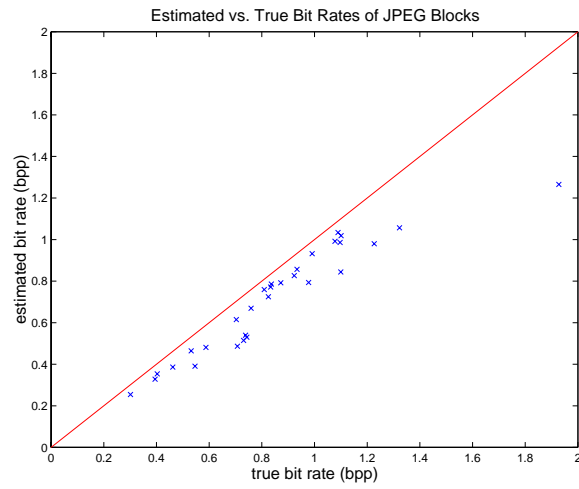




(a) One-color blocks



(b) Two-color blocks



(c) JPEG blocks

Figure 15: Estimated vs. true bit rates of coding each class