High Quality MRC Document Coding

Guotong Feng and Charles A. Bouman*
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907-1285
E-mail: {fengg, bouman}@ecn.purdue.edu
Phone: (765) 494–0340
Fax: (765) 494–3358

Abstract

The mixed raster content (MRC) model can be used to implement highly effective document compression algorithms. MRC document coders are typically based on the use of a binary mask layer that efficiently encodes the text and graphic content. However, while many MRC-based methods can yield much higher compression ratios than conventional color image compression methods, the binary representation tends to distort fine document details, such as thin lines and text edges.

In this paper, we propose a method for encoding and decoding the binary mask layer that substantially improves the decoded document fidelity of text and graphics at a fixed bit rate. This method, which we call resolution enhanced rendering (RER), works by adaptively dithering the encoded binary mask, and then applying a nonlinear predictor to decode a gray level mask at the same resolution. Both the dithering and nonlinear prediction algorithms are jointly optimized to produce the minimal distortion rendering. In addition, we introduce a second method, interpolative RER (IRER), which incorporates interpolation into the MRC decoder. The IRER method increases the compression ratio by allowing a high resolution document to be coded at lower resolution. We present experimental results illustrating the performance of our RER/IRER methods and comparing them to some existing MRC-based compression algorithms.

Primary EDICS: 4–IREP
Secondary EDICS: 4–DISP, 1–STIL

This work was supported by Xerox Foundation.

I. Introduction

A range of document imaging applications such as scan-to-print, document archiving, internet fax, and internet browsing are driving the need for document compression standards that maintain high quality while achieving high compression ratios. Without compression, a typical one page color document scanned at 600 dots per inch (dpi) requires approximately 90 Megabytes.

Conventional lossy image compression algorithms, such as JPEG or wavelet encoders [1], [2], [3], [4], are designed for the compression of natural images, and yield very poor quality/bit rate trade-offs for typical document images. This is particular true when the document contains a mixture of text, line art, pictures, and graphic content.

Over the last few years, the mixed raster content (MRC) model has been adopted as a standard for document encoding [5], [6] because it can dramatically improve image quality or reduce bit rate as compared to conventional picture encoders. MRC is an ITU standard (T.44) originally proposed for color facsimile [7]. It has also been proposed as the JPEG2000 architecture framework for the compound image file format [8]. The basic concept behind the MRC model is to separate a document into foreground and background layers using a binary mask. The binary mask then efficiently encodes the high spatial frequency information associated with text, while the foreground and background layers can be effectively compressed using conventional lossy image compression algorithms. A key concept in the T.44 implementation of MRC is that each of the three (or more) layers is compressed independently by a suitable encoder. This approach has the important advantages of simplicity and modularity, but at the cost of some coding overhead.

A number of MRC-based compression approaches have been proposed and studied in previous research [9], [10], [11], [12], [13], [14]. Among them, DjVu [9] and Digipaper [10] have already been employed in commercial products. In [11], de Queiroz *et al.* proposed a block-based MRC compression algorithm that relies on optimized block-thresholding segmentation in an approximate rate-distortion sense. In [15], Cheng *et al.* proposed a block-based multilayer compression algorithm using a rate-distortion optimized segmentation (RDOS) model. This algorithm was converted to a true MRC-based scheme in [13]. Closely related algorithms have been developed for check image compression in banking applications [16].

An intrinsic limitation of a standard MRC imaging model is that the binary mask can only represent discontinuous transitions between text, line art, and background colors. In practice, this type of hard transition can introduce substantial artifacts since real documents often contain subtle continuous-tone transitions between regions. These continuous transitions are useful in anti-aliasing, and allows the document to be encoded with lower dots per inch (dpi), which can further reduce bit rate. In principal,

it is possible to add edge detail to the foreground or background layers; however, in practice this detail is lost when these layers are subsampled and encoded using lossy natural image coders at acceptable bit rates.

In this paper, we propose a method called resolution enhanced rendering (RER) for jointly optimizing the MRC encoder and decoder to achieve low distortion rendering of edge transitions in the MRC binary mask layer [17]. The method works by adaptively dithering the mask layer of a three-layer MRC encoding to produce the intermediate tone levels required for low distortion rendering. The dithering is performed using a novel adaptive error diffusion algorithm. A tree-based nonlinear predictor is then designed into the MRC decoder to reconstruct the desired intermediate tones. Both the dithering and nonlinear prediction algorithms are jointly optimized to produce the minimal distortion rendering. The optimization is performed by iteratively optimizing the encoder and decoder to achieve the minimum distortion.

Based on this method, we will show how RER can also be used to interpolate text and graphics to a higher spatial resolution, while retaining the favorable bit rate of a lower resolution encoding. This can be done by interpolating the binary mask during the decoding process using a tree-based nonlinear interpolator. Since this interpolative RER method (IRER) produces higher resolution text and graphics, it allows the raster document to be represented at a lower spatial resolution, thereby reducing the overall bit rate.

The tree-based predictor and interpolator utilized in the RER and IRER methods are based on an image scaling approach called tree-based resolution synthesis (TBRS) proposed by Atkins *et al.* [18]. The TBRS algorithm differs from the traditional linear interpolation [19], [20], [21] and nonlinear interpolation techniques [22], [23], [24], [25], [26] in a number of important ways. The algorithm provides an effi cient classifi cation stage to separate distinct edge regions into different classes, each of which is then treated separately using an independently trained linear interpolator. Whereas linear interpolators tend to blur edges, the TBRS interpolation produces the sharp edge transitions necessary for our application.

An important advantage of RER/IRER is that it is compatible with the new standard MRC T.44 standard, and can be used to enhance the quality/bit rate trade-off of any conventional MRC encoder/decoder. This is because it encodes edge gradations directly into the spatial attributes of the binary mask.

This paper is organized as follows. In Section II the conventional MRC encoding schemes will be briefly described. In Section III, we will present the details of the proposed resolution enhanced rendering (RER) method for both MRC encoder and decoder, followed by the interpolative resolution enhanced rendering (IRER) algorithm. In Section IV, we will then develop the training procedure for the RER and

IRER parameters for both MRC encoder and decoder. Finally, we will present experimental results and conclusions in Section V and Section VI.

II. Conventional MRC Encoder

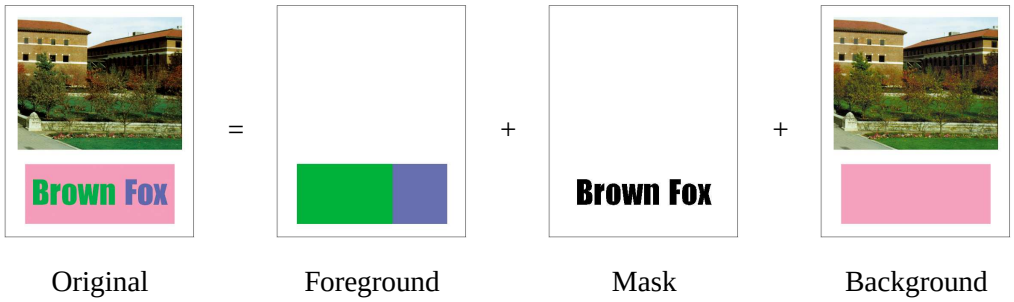


Fig. 1. MRC imaging model forms text and line art by using a binary mask to choose between foreground and background layers.

Fig. 1 illustrates a three-layer MRC document containing a background layer, a foreground layer, and a binary mask layer. At each pixel, the value of the binary mask is used to select between the foreground and background pixels in order to reconstruct the color raster image. Typically, the foreground layer contains the color of text and line drawings, and the background layer contains pictures, graphics and paper background. In general, both the foreground and the background layer can tolerate low spatial resolution but require high color resolution. On the other hand, the mask layer defines the shape of text and line graphics that require high spatial resolution.

In the MRC model, each layer is compressed independently. This adds some inefficiency since the foreground and background layers must be coded even when they are not used [27], but it simplifies the imaging model. Typically, the foreground and background layers are compressed at lower resolution using lossy natural image coders such as JPEG or JPEG2000 [1], [2]; whereas the binary mask is typically encoded with a lossless binary encoder such as CCITT Group 4, JBIG or JBIG2 [28].

In this paper, we focus on two MRC-based encoders for the implementation of our resolution enhancement rendering algorithm. One is the well known DjVu encoder [9], and the second one, referred to as RDOS-MRC (see details in Appendix), is an improved version of the MRC encoder proposed in [13]. Notice that DjVu is not strictly MRC-compliant because there exists some interaction between layers in the encoding procedure.

Both DjVu and RDOS-MRC can achieve high compression ratios for document coding while preserving good text readability. However, they both suffer the same problem as described in Section I. Fig. 2(b)

shows an example of DjVu decoded text. In the original scanned image as shown in Fig. 2(a), the anti-aliasing effect generated in the scanning process provides smoother and finer appearance along edges. Comparatively, as a result of losing the anti-aliasing effect, the DjVu decoded image exhibits objectional "jaggie" artifacts around the text edges. This problem is not desirable in high quality MRC document coding.

III. Resolution Enhanced Rendering Method

Fig. 2(c) illustrates how the resolution enhanced rendering (RER) algorithm functions. First, the RER encoder segments the foreground and background using an adaptive error diffusion method. This error diffusion method effectively dithers the binary mask along the edges of the characters to represent the gradual transition of the scanned text characters. The RER decoder then uses this binary mask, together with the foreground and background colors to estimate the true value of the document pixels. This estimation is done using a nonlinear tree-structured predictor as described in [29], [18]. Importantly, this predictor is trained to identify the characteristic patterns of the RER encoder. Therefore, it can do a much better job of accurately estimating the true pixel values.

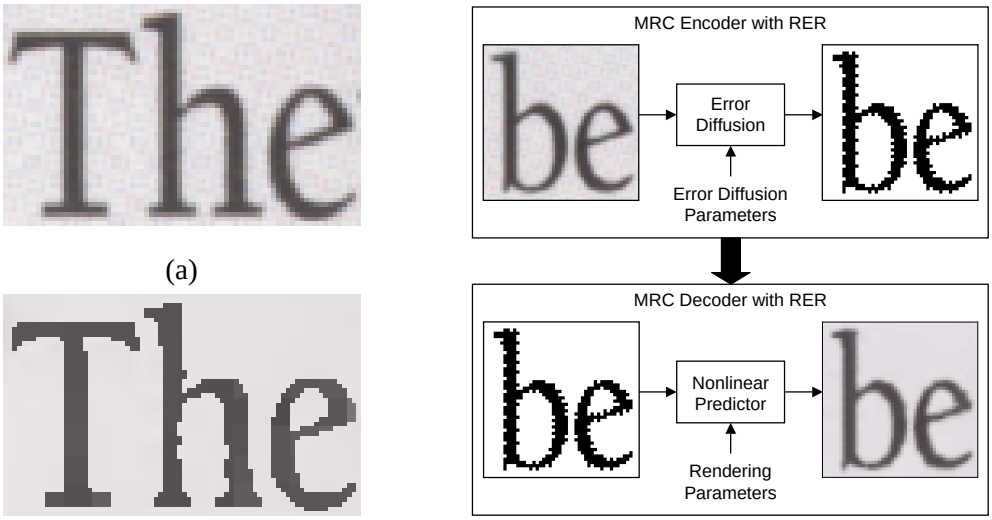


Fig. 2. Illustration of conventional MRC and RER enhanced MRC. (a) Original image scanned at 400 dpi. (b) Decoded image compressed by DjVu at 0.158 bpp (152:1 compression ratio). (c) Illustration of MRC encoder and decoder with RER. Examples were selected from actual RER inputs and outputs.

Fig. 3 illustrates how the RER encoder and decoder are jointly optimized to minimize distortion of the decoded document. As we will see, both the encoder and the decoder have parameters which can be trained to produce the best possible result. The error diffusion algorithm has five parameters which

control its behavior, and the nonlinear predictor has a large number of parameters which specify the nodes of a nonlinear regression tree.

In each iteration of the optimization, the parameters of the encoder or decoder are alternatively fixed, while the parameters of the other one are optimized. Importantly, two disjoint sets of documents are used for training the encoder and decoder. We have found that this improves the robustness of the training procedure. Experimental results are shown for test documents that are not contained in either set of training documents. The experimental results indicate that this training process robustly converges to parameters which reduce the distortion of the decoded document. Moreover, we have found that joint optimization of the encoder and decoder performs substantially better than independent optimization of these two functions.

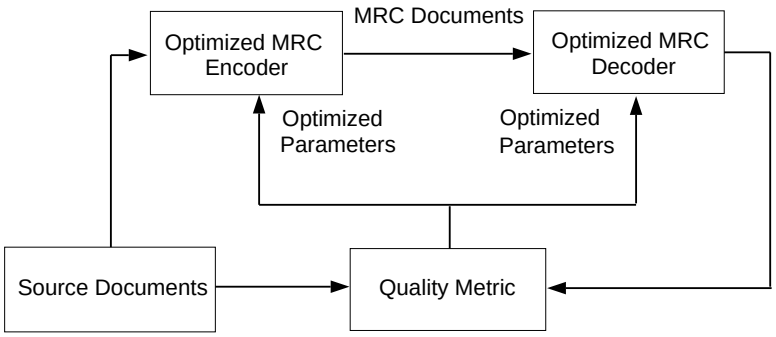


Fig. 3. Overview of method used to train the optimized encoder and decoder. Once training is complete, the encoder and decoder function independently.

A. RER Encoder

Fig. 4(a) illustrates the structure of a typical 3-layer MRC encoder, and Fig. 4(b) illustrates the basic structure of an RER encoder. For a typical 3-layer MRC encoder, we assume that the foreground and background layers are compressed at lower resolutions than the original by using a lossy continuous-tone encoder, and the binary mask layer is compressed at the original resolution by using a lossless binary image encoder. The resolutions and the quality levels of encoding for the foreground and background layers may or may not be the same. In the RER encoder, the RER encoding module takes the foreground layer, background layer, binary mask layer, and original document as the input, and creates a dithered mask layer. The dithered mask, foreground, and background layers are then separately compressed by using the binary image encoder and continuous-tone encoder that are used for the MRC encoder. Let X_s be the pixel color in the raster document at location s. Let F_s and B_s be the decimated foreground and

background colors at location s. Next, define M_s as the binary mask, and D_s as the dithered mask. Let $\tilde{F}_s$ and $\tilde{B}_s$ be the interpolated foreground and background colors at location s.

To compute the dithered mask, first, a simple edge detection procedure is performed on the binary mask layer to find pixels that lie on the boundary between foreground and background. Let K_s be the binary output of the boundary detection procedure. If K_s is 1, s is a boundary pixel. Otherwise, it is a non-boundary pixel. Next, the foreground layer, F_s, and the background layer, B_s, are interpolated back to the original resolution to form $\tilde{F}_s$ and $\tilde{B}_s$. Then, at each boundary pixel, the original color is projected onto the line connecting the associated foreground and background colors (i.e. $\tilde{F}_s$ and $\tilde{B}_s$), creating a real value that represents the relative mixture of foreground and background colors at that pixel, defined as λ_s. Finally, an adaptive error diffusion procedure is switched on at each boundary pixel by taking λ_s as the input, and creating the output, D_s. At each non-boundary pixel, the error diffusion procedure is switched off and the binary mask value, M_s, is simply copied to the output, D_s. Thus a binary output image of the dithered mask has been produced.

We refer to λ_s as the gray level mask. More specifically, λ_s is given by the value on the real line which minimizes the squared error $||X_s - (\tilde{B}_s + \lambda_s(\tilde{F}_s - \tilde{B}_s))||^2$. Fig. 5 gives a geometric interpretation of λ_s as the projection of the true pixel color onto the line connecting the foreground and background colors. The solution to this least squares approximation problem is given by

$$\lambda_s = \frac{(X_s - \tilde{B}_s)^t(\tilde{F}_s - \tilde{B}_s)}{(\tilde{F}_s - \tilde{B}_s)^t(\tilde{F}_s - \tilde{B}_s)} \ . \tag{1}$$

Furthermore, let γ_s denote the value of λ_s constrained to the interval $[0, 1]$.

$$\gamma_s = \min\{1, \max\{0, \lambda_s\}\} \ . \tag{2}$$

So, γ_s forms a gray scale image with minimum value 0 and maximum value 1. Notice that in the boundary case, when $\gamma_s = 0$, the pixel s is primarily background, and when $\gamma_s = 1$, the pixel s is primarily foreground.

We also define the minimum approximation error at each pixel, Δ_s, in terms of the value γ_s.

$$\Delta_s = \min\{\gamma_s, 1 - \gamma_s\} \ . \tag{3}$$

The value of Δ_s falls in the interval $[0, 0.5]$. Notice that when a pixel is well approximated by either the foreground or background color, then Δ_s is small. On the other hand, when a pixel is best approximated by a mixture of foreground and background colors, then Δ_s is large. The value of Δ_s will be used to control the local adaptation of the error diffusion algorithm.

The RER encoder computes the binary mask by applying a form of adaptive error diffusion to the gray scale image γ_s. While the RER error diffusion method is similar to serpentine scan Floyd Steinberg

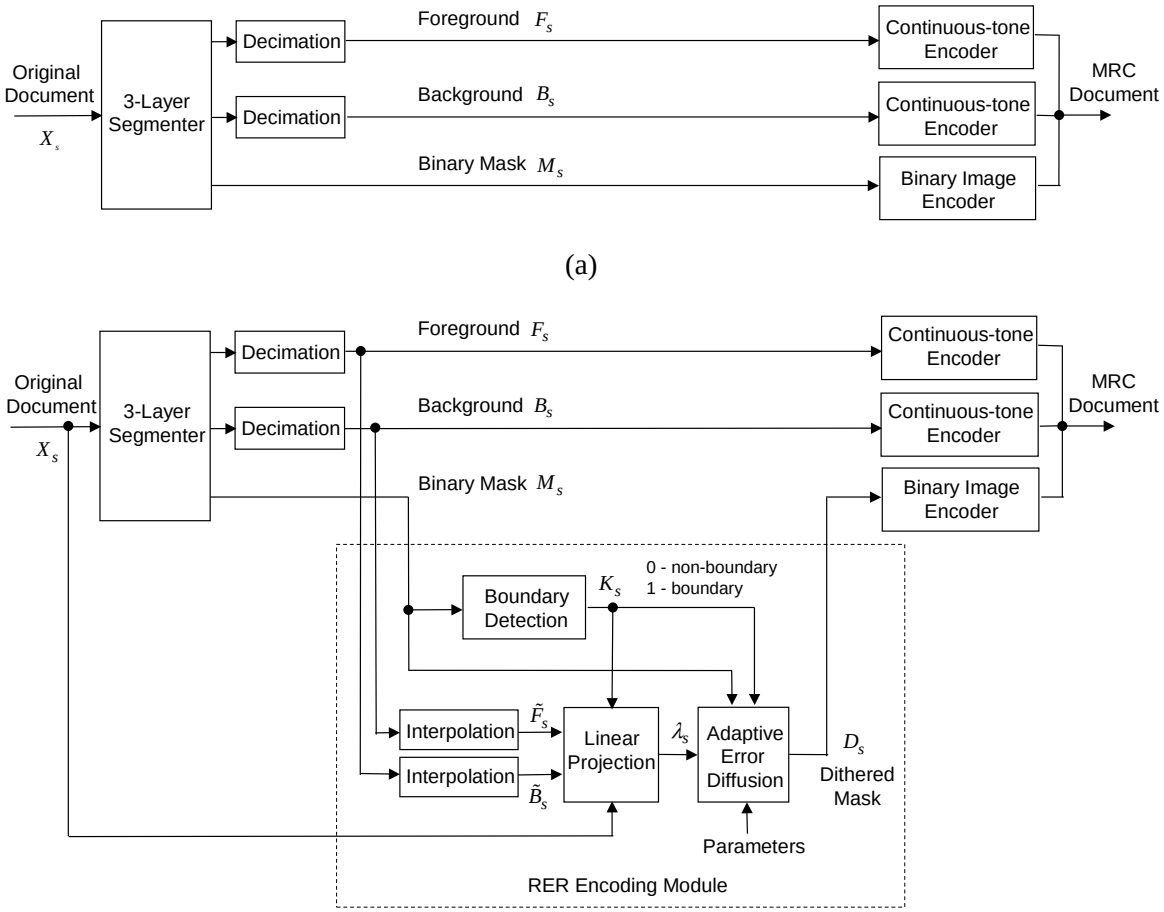


Fig. 4. (a) Structure of a typical 3-layer MRC encoder. (b) Basic structure of an RER enhanced MRC encoder.

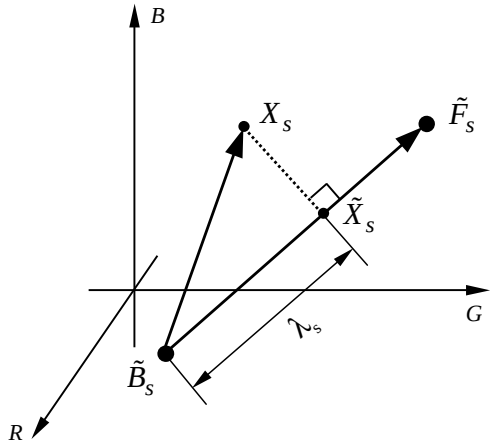


Fig. 5. Least squares approximation ($\tilde{X}_s$) of pixel color, X_s, by a combination of the background, $\tilde{B}_s$, and foreground, $\tilde{F}_s$, colors.

error diffusion [30], it is specially designed and optimized to diffuse error along text edge transitions. This is done by adaptively setting the error diffusion weights at each pixel.

Fig. 6 illustrates the four future pixels s_0, s_1, s_2, s_3 which are the neighbors of s when the scan order of error diffusion is from left to right. Similarly, the positions of the four future pixels are flipped over horizontally about pixel s when the scan order is from right to left. The values $w_{s_0}, w_{s_1}, w_{s_2}, w_{s_3}$ correspond to the four error diffusion weights for the pixel s. The values of these four weights are varied at each pixel s using the formula

$$w_{s_j} = \frac{\alpha_j \Delta_{s_j} u(\Delta_{s_j} - \tau)}{\sum_{j=0}^{3} \alpha_j \Delta_{s_j} + 0.001} \ , \tag{4}$$

where Δ_{s_j} is the approximation error at s_j defined in (3), $u(\cdot)$ is the unit step function, and $(\tau, \alpha_0, \alpha_1, \alpha_2, \alpha_3)$ is a five dimensional vector of scalar parameters. The value of τ falls in the interval $[0, 0.5]$, and the values of the other four parameters fall in the interval $[0, 1]$.

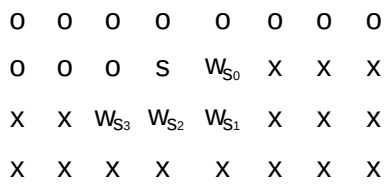
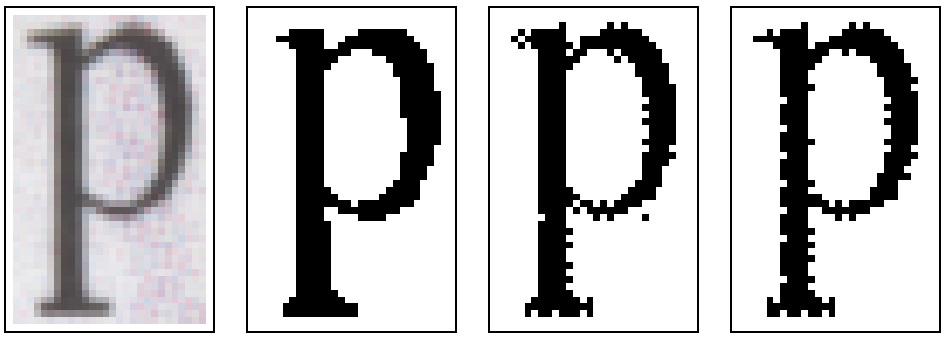


Fig. 6. Illustration of adaptive error diffusion algorithm. The o's are pixels that have already been processed, the current pixel position is denoted by s, and s_0, s_1, s_2, s_3 denote the four future pixel positions to which error will be diffused. The corresponding error diffusion weights at these four positions are denoted by $w_{s_0}, w_{s_1}, w_{s_2}, w_{s_3}$. These four weights are adapted based on local edge orientation in the document image.

Equation (4) is designed to diffuse the error in the direction with the largest approximation error. In the case of text edges, this means that it tends to diffuse the error along the edge of the character. For example, if the text edge is strictly horizontal, w_{s_0} becomes dominant and the other three weights become insignificantly smaller, thereby encouraging the error at pixel s to be mostly diffused to its horizontal neighbor s_0. Moreover, when the parameter $\tau > 0$, the error is not diffused into regions where the approximation is good, since in this case $u(\Delta_{s_j} - \tau) = 0$. This means that the error is not generally diffused into smooth interior regions of a character where it could generate isolated holes in the binary mask.

Fig. 7 shows an example of how this dithering procedure works. Notice that the binary mask without any dithering does not contain adequate local edge information, and Floyd Steinberg error diffusion does not generate the desired error distribution along the edge. In comparison, the dithered binary mask demonstrates that Equation (4) enables a directionally adaptive error diffusion procedure, which produces a homogeneous dither pattern along the edge.



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

Fig. 7. Example of binary mask dithering. (a) A portion of original image scanned at 400 dpi. (b) Binary mask without dithering. (c) Binary mask dithered with Floyd Steinberg error diffusion. (d) Binary mask dithered with directionally adaptive error diffusion, the four error diffusing weights of which are controlled by Equation (4).

This RER dithering procedure is controlled by the parameter vector $\theta = (\tau, \alpha_0, \alpha_1, \alpha_2, \alpha_3)$. Section IV describes how this parameter vector is estimated from training data to minimize decoded document distortion.

B. RER Decoder

Fig. 8(a) illustrates the structure of a typical 3-layer MRC decoder, and Fig. 8(b) shows the basic structure of how an RER decoder works. We assume that the decoded foreground and background layers have lower resolution than the original, and any interpolation algorithm can be applied for interpolating the two layers. Let $\tilde{F}_s$ and $\tilde{B}_s$ be the decoded foreground and background layers that have been interpolated, respectively, both having the same resolution as the original. First, the binary mask, M_s, is input to a boundary detection module, which works exactly the same way as in the RER encoder. For each non-boundary pixel, the reconstructed pixel color, denoted as $\hat{X}_s$, is chosen between the foreground and background colors (i.e. $\tilde{F}_s$ and $\tilde{B}_s$) according to the binary mask, the same way as in a standard MRC decoder. However, if the pixel is on the boundary, then a nonlinear prediction algorithm is performed to render the reconstructed pixel color. The nonlinear predictor works by computing, $\hat{\lambda}_s$, a minimum mean squared error estimate of λ_s. Using this estimate, the reconstructed pixel color can be computed as

$$\hat{X}_s = \hat{\lambda}_s \tilde{F}_s + (1 - \hat{\lambda}_s)\tilde{B}_s \ . \tag{5}$$

Fig. 9 shows the structure of the nonlinear predictor that contains a tree-based classifier and a set of class-dependent least square estimators. The nonlinear predictor works by first extracting the binary mask in a 5×5 window about the pixel in question. This data forms a binary vector, z_s, which is then used as input to a binary regression tree predictor known as tree-based resolution synthesis (TBRS) [29],

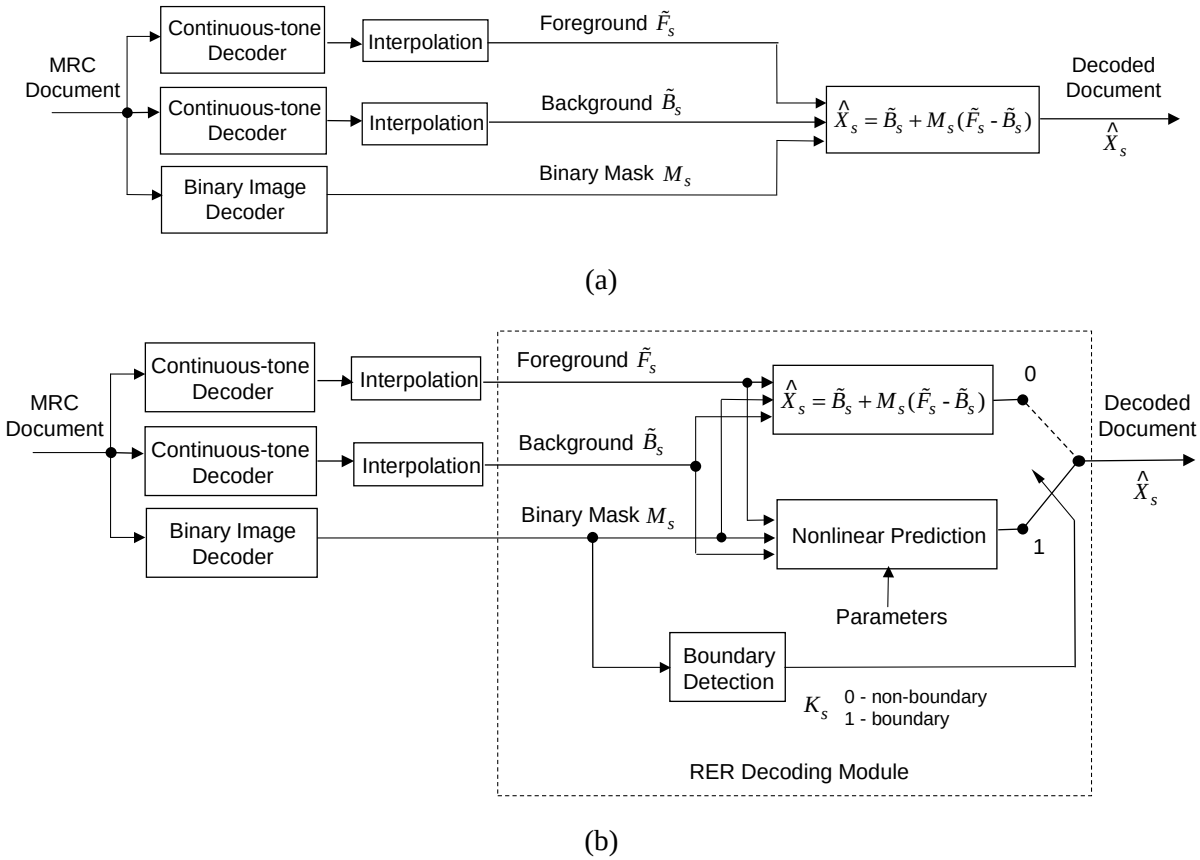


Fig. 8. (a) Structure of a typical 3-layer MRC decoder. (b) Basic structure of an RER decoder.

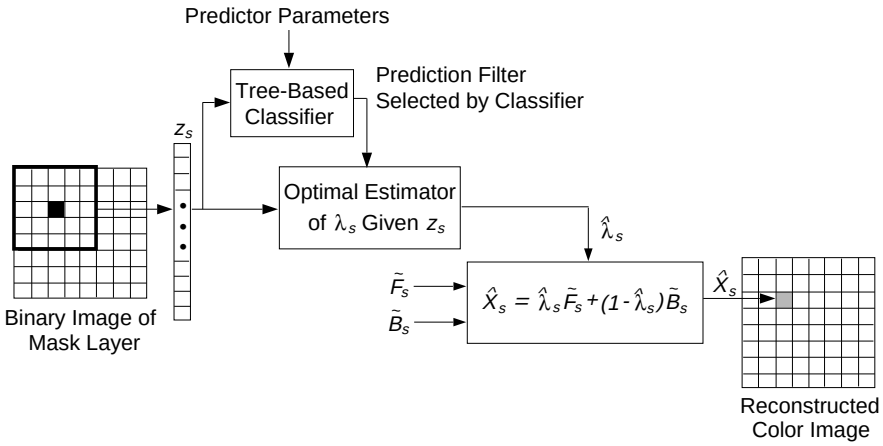


Fig. 9. Structure of RER predictor using tree-based nonlinear estimation.

[18]. The TBRS predictor estimates the value of λ_s in a two-step process, classification and prediction. First, it classifies the vector z_s into one of M classes using a binary tree classifier. Each class, then has a

corresponding linear prediction filter which is used to estimate the value of λ_s from z_s using the equation

$$\hat{\lambda}_s = A_m z_s + \beta_m \ , \tag{6}$$

where m is the determined class of the vector z_s, A_m and β_m are the corresponding linear prediction parameters of class m.

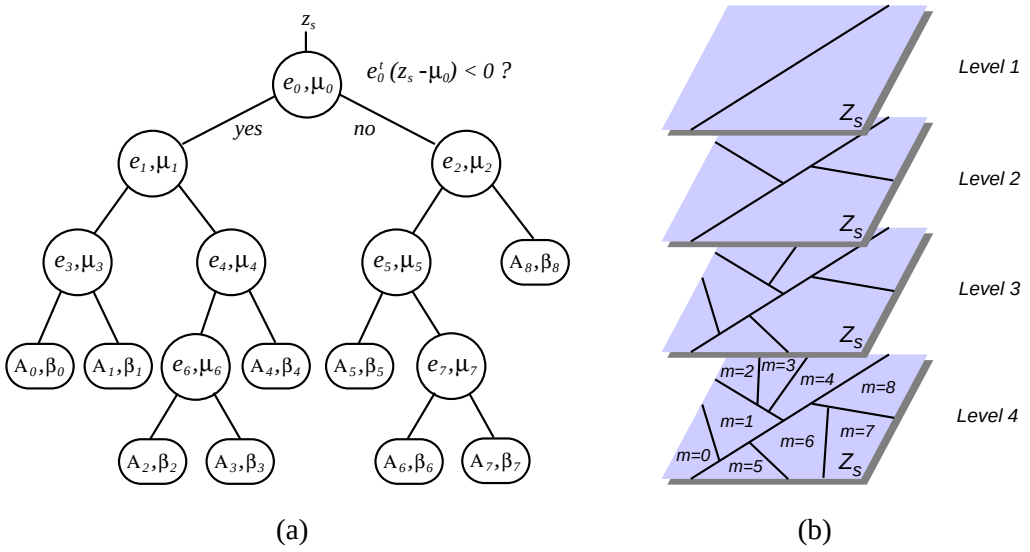


Fig. 10. Illustration of the binary tree prediction model. (a) The regression tree structure. e_i and μ_i specify the binary decision rule for splitting node i. A_m and β_m are the linear filter parameters of least square estimation for leaf (or class) m. (b) The recursive classification procedure of the binary tree in the input vector space Z_s.

Fig. 10 illustrates the structure of the binary tree prediction model and its classification procedure. As shown in Fig. 10(a), the input vector z_s is first classified by traversing down the binary tree from the root at the top, making a binary decision at each non-terminal node that the input vector passes through, and terminating at one of the leaves at the bottom, where each leaf represents a distinct class of the input vector space Z_s. At each non-terminal node i, the input vector is determined to go to the left child or the right child using the specific splitting rule

$$e_i^t(z_s - \mu_i) \quad \begin{matrix} \geq \\ < \end{matrix} \quad 0 \ , \tag{7}$$

where e_i is a pre-computed vector specifying the orientation of a hyperplane for splitting, e_i^t is the transpose of e_i, and μ_i is a reference point on the hyperplane. Specifically, if the projection of $z_s - \mu_i$ onto e_i is negative, then z_s falls into the left side of the hyperplane passing through the reference point μ_i and perpendicular to the vector e_i and consequently goes down to the left child of node i. Otherwise, it falls to the right side of the hyperplane and goes down to the right child. Both e_i and μ_i were obtained from the training process, which will be described in Section IV.

At the bottom of the tree, each leaf (i.e. terminal node) denoted by m represents a distinct class of Z_s, containing a set of linear prediction parameters A_m, β_m. If z_s is classified into class m, then the linear prediction of λ_s given z_s can be computed using Equation 6. Thus the entire set of linear prediction filters for all the classes provide a piecewise approximation of the nonlinear conditional mean estimator. Fig. 10(b) shows an example of the recursive classification procedure in the input vector space Z_s. At the bottom level, each polygon region represents a different class of Z_s. The classification step is essential because it can separate out the distinct regions of the document corresponding to mask edges of different orientation and shape.

C. The Interpolative RER (IRER) Decoder

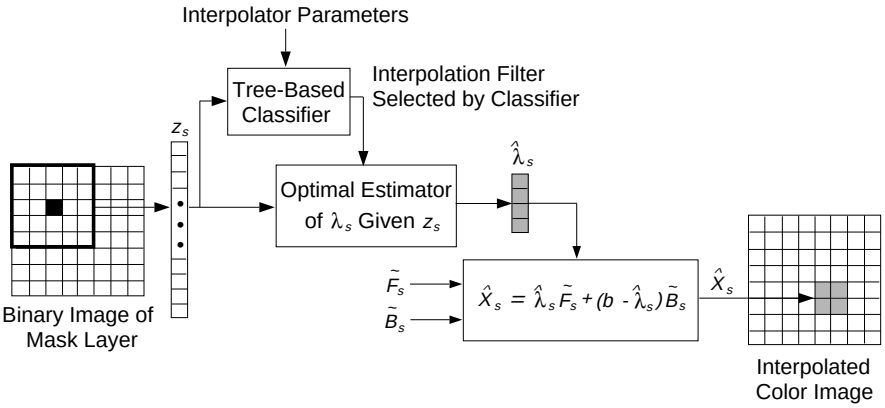


Fig. 11. Structure of IRER interpolator using tree-based nonlinear interpolation.

With the same encoder, the RER decoder can be extended to interpolate the binary mask. We call this the interpolative RER (IRER) decoder. In the IRER decoder, the estimated gray level mask $\hat{\lambda}_s$ has twice the sample resolution of the original gray level mask λ_s. This is done by designing the nonlinear predictor to estimate four output values for each input pixel location. Fig. 11 illustrates the structure of the IRER interpolator. This interpolator has the same structure as used in the RER decoder, except that the prediction filters A_m and β_m are designed to interpolate the gray level mask $\hat{\lambda}_s$, which is now a vector of 4 elements. Using this interpolation algorithm, the reconstructed pixel color is computed as

$$\hat{X}_s = \hat{\lambda}_s \tilde{F}_s + (b - \hat{\lambda}_s)\tilde{B}_s \ , \tag{8}$$

where $b = [1, 1, 1, 1]^t$, $\hat{X}_s$ and $\hat{\lambda}_s$ are 4-dimensional vectors, and $\tilde{F}_s$ and $\tilde{B}_s$ are scalar values, referred to as the decoded foreground and background layers with the same resolution as the original.

Fig. 12 shows a typical example of an imaging pipeline for rendering of a 600 dpi document using the IRER technique. In this imaging pipeline, the 300 dpi original raster image is compressed with an

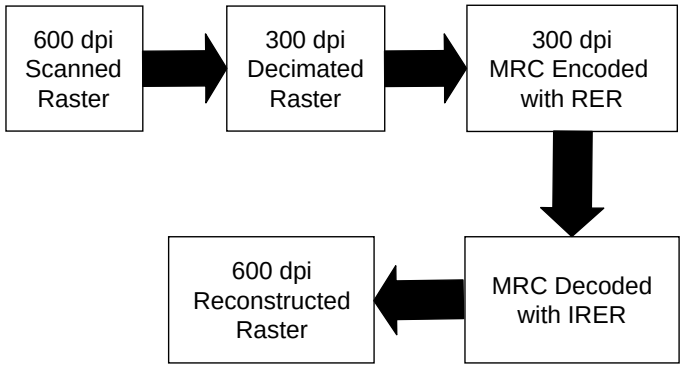


Fig. 12. Example of imaging pipeline using the interpolative RER decoding technique.

RER encoder, and then decompressed with an IRER decoder, producing a 600 dpi reconstructed raster image. With the effi cient interpolation of the RER encoded gray level mask, the high resolution text edges and shape can be accurately reconstructed, while the bit rate of encoding the original document at high resolution can be substantially reduced.

IV. Training

The objective of the training process is to optimize the performance of the RER encoder and decoder by selecting the encoder and decoder parameters to minimize the decoded document distortion over a training set of documents. The distortion metric used to measure document quality is mean squared error. While mean squared error is not always a good measure of quality, in this application we found it to be well correlated with our subjective evaluation of document quality [15].

The training process alternates between optimization of the encoder and decoder parameters, and the iteration is always started by optimizing the decoder. When optimizing the encoder parameters, the previously obtained decoder parameters are used; and when optimizing the decoder parameters, the previously obtained encoder parameters are used. Importantly, the training phases for encoder and decoder used different sets of training data. This strategy was found to produce more robust training results.

A. Decoder Optimization

While the TBRS predictor is simple to implement, it can be computationally expensive to train [18], [29], especially when a relatively large set of image training data is necessary for desired prediction and interpolation performance. The ultimate goal of the training process is to generate the optimal binary regression tree that best approximates the conditional mean estimator. To do this, the shape and size of the regression tree are optimized by performing a tree growing phase followed by a tree pruning phase,

each phase using a different set of training data. Sample mean squared interpolation errors are computed to evaluate the quality of the candidate trees generated during the two phases. When this growing-then-pruning process is complete, a large third set of training data is applied to the resulting regression tree to compute the new interpolation filter parameters for each leaf (i.e. each class), yielding the final TBRS predictor. Fortunately, although the entire training process may be computationally demanding, we only need to conduct the training once.

1) Generating training sets: The training sets are obtained by selecting relevant pixels from a wide variety of training document images containing mixed content. At each selected pixel, we extract the values of λ_s and z_s, the binary vector representing the mask values in a 5×5 window about the pixel s. In the case of RER, λ_s is only a scalar value that represents the mixture of foreground and background color at the pixel s. In the case of IRER, where the interpolation factor is 2, λ_s is a vector representing the mixture values of foreground and background color for the high-resolution pixels in a 2×2 window corresponding to the low-resolution pixel s. These training pairs, (λ_s, z_s), are then used to train the predictor.

The training pairs for RER are only extracted at pixels for which $-0.1 \leq \lambda_s \leq 1.1$. Similarly, the training pairs for IRER are only extracted at pixels for which each element of the vector λ_s falls in the interval $[-0.1, 1.1]$. We found that selecting training pairs in this manner substantially reduced the distortion of the decoded documents.

We generate three sets, G, P and D, from all the extracted training pairs. The sets G and P are disjoint, and are used for the tree growing and tree pruning stages, respectively. D is a much larger set than G and P which is used for generating the final prediction filters. We also found that it was important for G and P to be extracted at different pixels from spatially separate regions containing a combination of foreground and background content. The training set D should also represent a wide variety of document image data containing a combination of foreground and background content.

2) Tree growing and pruning: Tree growing is a top-to-bottom recursive splitting procedure at a sequence of nodes that yield the largest reduction in the sample mean squared prediction error. This is done by splitting all the candidate nodes in each step and selecting one of them as the splitting node that can result in the largest reduction of its prediction error. The splitting procedure at each node is based on a multivariate splitting strategy, which is illustrated in Fig. 13.

Let i be the node being split. Assume that there are N samples of the training pairs that fall into node i, where λ_{s_n} and z_{s_n} $(n = 1, \ldots, N)$ are the samples of Λ_s and Z_s, separately. We then define two

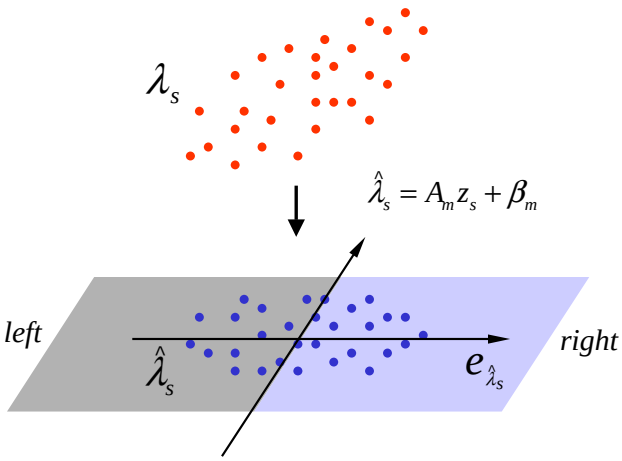


Fig. 13. Illustration of multivariate splitting procedure. $\hat{\lambda}_s$ is the least square estimation of λ_s given z_s. $e_{\hat{\lambda}_s}$ is the principle eigenvector of a covariance matrix estimate of $\hat{\lambda}_s$. $\hat{\lambda}_s$ is divided into the left and right set by a hyperplane perpendicular to $e_{\hat{\lambda}_s}$.

matrices Λ and Z as

$$\begin{aligned} \Lambda &= [\lambda_{s_1} - \hat{\mu}_{\lambda_s}, \lambda_{s_2} - \hat{\mu}_{\lambda_s}, \ldots, \lambda_{s_N} - \hat{\mu}_{\lambda_s}] \\ Z &= [z_{s_1} - \hat{\mu}_{z_s}, z_{s_2} - \hat{\mu}_{z_s}, \ldots, z_{s_N} - \hat{\mu}_{z_s}] \ , \end{aligned}$$

where $\hat{\mu}_{\lambda_s}$ and $\hat{\mu}_{z_s}$ are the sample mean estimates of λ_s and z_s, respectively, computed as

$$\begin{aligned} \hat{\mu}_{\lambda_s} &= \frac{1}{N}\sum_{n=1}^{N} \lambda_{s_n} \\ \hat{\mu}_{z_s} &= \frac{1}{N}\sum_{n=1}^{N} z_{s_n} \ . \end{aligned}$$

Using least square estimation, the linear prediction filters A_i and β_i of node i are computed as

$$A_i = \Lambda Z^t (Z Z^t)^{-1} \tag{9}$$

$$\beta_i = \hat{\mu}_{\lambda_s} - \Lambda Z^t (Z Z^t)^{-1} \hat{\mu}_{z_s} \ . \tag{10}$$

We then obtain the covariance estimate $\hat{R}_{\hat{\lambda}_s}$ as

$$\hat{R}_{\hat{\lambda}_s} = A_i Z Z^t A_i^t \ . \tag{11}$$

Let $e_{\hat{\lambda}_s}$ be the principle eigenvector of $\hat{R}_{\hat{\lambda}_s}$. The desired splitting rule is then defined as

$$e_i^t (z_{s_n} - \hat{\mu}_{z_s}) \ \gtrless \ 0 \ , \tag{12}$$

where $e_i^t = e_{\hat{\lambda}_s}^t A_i$. Specifically, if $e_i^t (z_{s_n} - \hat{\mu}_{z_s}) < 0$, z_{s_n} goes to the left child of node i; otherwise, z_{s_n} goes to the right. Notice that

$$e_i^t (z_s - \hat{\mu}_{z_s}) = e_{\hat{\lambda}_s}^t (\hat{\lambda}_s - \hat{\mu}_{\hat{\lambda}_s}) \ , \tag{13}$$

where $\hat{\mu}_{\hat{\lambda}_s} = A_i \hat{\mu}_{z_s}$ is an estimate of the mean of $\hat{\lambda}_s$. This equation demonstrates that the splitting rule for z_s is essentially equivalent to dividing $\hat{\lambda}_s$ with a hyperplane passing through its mean estimate and perpendicular to the direction along which $\hat{\lambda}_s$ varies the most.

The recursive splitting procedure will terminate when the leaves of the tree grow to a predefined number M_{max}. Usually this number is selected to be large enough so that the resulting tree over-represents the growing data set, G. Next, a tree-pruning stage is started to refine the tree by eliminating extraneous branches using a different training set P. This is done by performing a bottom-to-top recursive procedure in which, starting from the terminal nodes, each pair of children is compared to their parent in terms of the mean-squared prediction error. If there is a reduction in the prediction error at a parent, all the subtrees of its children are pruned. This procedure is recursively repeated up toward the root, generating a smaller tree structure which tends to represent the observable distinct classes in an efficient way.

3) Retraining prediction filters: To further improve the predicting performance for the tree obtained from the pruning stage, a considerably larger training set D is used to retrain the prediction filter in each leaf. Specifically, all the samples of set D are first classified into different leaves by traversing down the regression tree from its root. Then least square estimation is applied to the data cluster in each leaf, generating new parameters of the prediction filter.

B. Encoder Optimization

The encoder is optimized by searching for the value of the vector $\theta = [\tau, \alpha_0, \alpha_1, \alpha_2, \alpha_3]$ which minimizes the mean squared error averaged over the set of training documents. This search is initialized at the current value of the vector θ and uses the decoder parameters resulting from the last optimization of the decoder.

More specifically, let U be the training set of original documents for the encoder optimization, and $\hat{U}_\theta$ be the corresponding set of decoded documents. Define $D(\cdot)$ as the quality metric (i.e. mean squared error) function. Then the optimization problem of the RER encoder can be written as

$$\hat{\theta} = \underset{\mathbf{0} \le \theta \le \mathbf{c}}{\arg\min}\, D(U, \hat{U}_\theta) \ , \tag{14}$$

where c is the upper limit of θ, which is $[0.5, 1, 1, 1, 1]^t$.

The optimization of θ is done by sequentially perturbing the elements of the vector θ using an iterative coordinate decent strategy. Each perturbation is made using a step size of $\pm\delta$ which is initialized to the value $\delta = 0.2$. The decision to perturb each component by $\pm\delta$ or to leave it unchanged is made based on the value of the mean squared error computed for the documents in the training set U. If the mean squared error does not decrease with the perturbation of every element of the parameter vector, then

the size of the perturbation is automatically reduced using the formula $\delta \leftarrow \delta/2$. Coordinate descent optimization is stopped when the perturbation value is less than 0.01.

V. Experimental Results

In this section, we present experimental results by applying the resolution enhanced rendering (RER) method and the interpolative enhanced resolution rendering (IRER) method to the RDOS-MRC and DjVu compression algorithms. All the results were produced from two test images shown in Fig. 14, each at two different resolutions, 400 dpi and 600 dpi. Test image I was scanned at both 400 dpi and 600 dpi on a Linotype-Hell (SAPHIR ultra 2) flatbed scanner, and test image II was synthesized at both 400 dpi and 600 dpi from an Adobe Illustrator original document using letter-sized format.

(a) (b)

Fig. 14. Test images used for RER and IRER. (a) Test image I, scanned at 400 dpi and 600 dpi. (b) Test image II, synthesized at 400 dpi and 600 dpi.

A. RER implementation for DjVu and RDOS-MRC

The implementation for RER enhanced DjVu is based on the latest free code (version 3.5.12) released on www.djvuzone.org. Since this free package does not include the segmentation module, the DjVu binary mask was generated from a DjVu segmenter available in an old version of the DjVu package (DjVu-0.9). DjVu is not fully MRC-compliant because not every pixel in the foreground and background is encoded and decoded. However, the RER method requires some of these missing foreground/background

pixels that fall along mask boundaries. In order to solve this problem, we extrapolated missing foreground/background pixels along boundaries by computing the average of available pixels of the same type in an 8 pixel neighborhood.

In the RER encoder, we used pixel replication for the foreground and background interpolation. In the RER decoder for RDOS-MRC, pixel replication was also used for the foreground and background interpolation; whereas in the RER decoder for DjVu, we directly adopted the interpolation algorithm that is provided by the DjVu software.

B. Training results

For the training process of RER and IRER, we used a training set of 12 documents, containing 9 scanned images and 3 synthetic images. All scanned images were scanned at both 400 dpi and 600 dpi, 24 bits per pixel (bpp), on the Linotype-Hell flatbed scanner, and all synthetic images were synthesized at both 400 dpi and 600 dpi, 24 bits per pixel (bpp), by converting a set of computer generated PDF documents using Adobe PhotoShop. All 400 dpi documents were used for the RER training, and all 600 dpi documents were used for the IRER training. Among the 12 documents, 2 documents were used for encoder optimization, including a scanned image and a synthetic image, and the other 10 documents were used for decoder optimization, including 8 scanned images and 2 synthetic images. The training set did not include the two test images.

The training process was performed using the RER enhanced RDOS-MRC encoder and decoder. For both RER and IRER, the joint optimization process of the encoder/decoder pair was ended after 5 iterations. The maximum number of the decoder prediction tree leaves was set to 1000 . The encoder parameter vector θ was initialized as $\theta = [\tau, \alpha_0, \alpha_1, \alpha_2, \alpha_3] = [0.25, 0.4375, 0.0625, 0.3125, 0.1875]$, where $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ were set to the four Floyd Steinberg error diffusion weights.

Table I lists the optimized results of the RER and IRER parameters using the scanned and synthesized training sets. In this table, the last two columns show the numbers of training samples that were used for the growing and pruning processes of the prediction tree, and the final numbers of the tree leaves at the end of optimization.

C. Results of RER

For all RDOS-MRC based methods, both background and foreground were encoded at 200 dpi (i.e. half the resolution of the original), and a fixed segmentation was used for all quality levels. For all DjVu based methods, the foreground and background were encoded at 67 dpi (i.e. 1/6 of the original) and 133

TABLE I

TRAINING RESULTS OF RER AND IRER PARAMETERS.

Method	Encoder					Decoder	
	τ	α_0	α_1	α_2	α_3	# of Samples (million)	# of Leaves
RER	0.2250	0.2250	0.0625	0.1625	0.0750	4.69	875
IRER	0.2375	0.2375	0.0500	0.2500	0.0875	3.09	908

dpi (i.e. 1/3 of the original), respectively. Fig. 15 shows the 400 dpi binary masks of test image I and II generated from DjVu and RDOS-MRC.

Fig. 16 shows the comparison among the original image, the images rendered by DjVu and RDOS-MRC with and without RER. All these results are taken at a bit rate of approximately 0.15 bits per pixel (bpp). Fig. 16(b) and Fig. 16(f) exhibit similar objectionable "jaggie" artifacts around the text edges, which is the common problem existing in MRC encoders. Fig. 16(c) and Fig. 16(g) show the RER encoded binary masks for DjVu and RDOS-MRC, and Fig. 16(e) and Fig. 16(i) show the corresponding RER decoded results for DjVu and RDOS-MRC. As observed in Fig. 16(e) and Fig. 16(i), the "jaggie" artifacts are eliminated and the overall visual quality is fairly close to the original shown in Fig. 16(a). As a comparison, an example of least squares linear prediction is shown in Fig. 16(d) and Fig. 16(h). The result was obtained by simply using the root node of the prediction tree of RER decoder as a single linear predictor. Compared to Fig. 16(e) and Fig. 16(i), the results in Fig. 16(d) and Fig. 16(h) appear more blurry and noisy around edges. This demonstrates that the RER nonlinear predictor can provide accurate anti-aliasing on the edges while still maintaining sharpness. Fig. 17 shows a similar example to Fig. 16, but for the synthetic test image II.

Fig. 18 illustrates the comparison of rate-distortion performance among DjVu, RER enhanced DjVu, RDOS-MRC, and RER enhanced RDOS-MRC, on both the scanned and synthesized images. The mean squared error per pixel per color was taken as the distortion measure for the R-D performance. The R-D curves of DjVu and RER enhanced DjVu were obtained by changing the quality level of wavelet compression from 5 to 100. The R-D curves of RDOS-MRC and RER enhanced RDOS-MRC were computed by changing the JPEG quality level from 3 to 90 for both foreground and background.

In Fig. 18(a), the rate-distortion curves of both the RER enhanced DjVu and RER enhanced RDOS-MRC show the substantial improvement achieved by the RER method. Each curve has a sharp knee at about 0.06 bits per pixel (400:1 compression ratio) and 0.11 bits per pixel (218:1 compression ratio) separately, and then remains nearly parallel to the standard DjVu and RDOS-MRC R-D curve separately

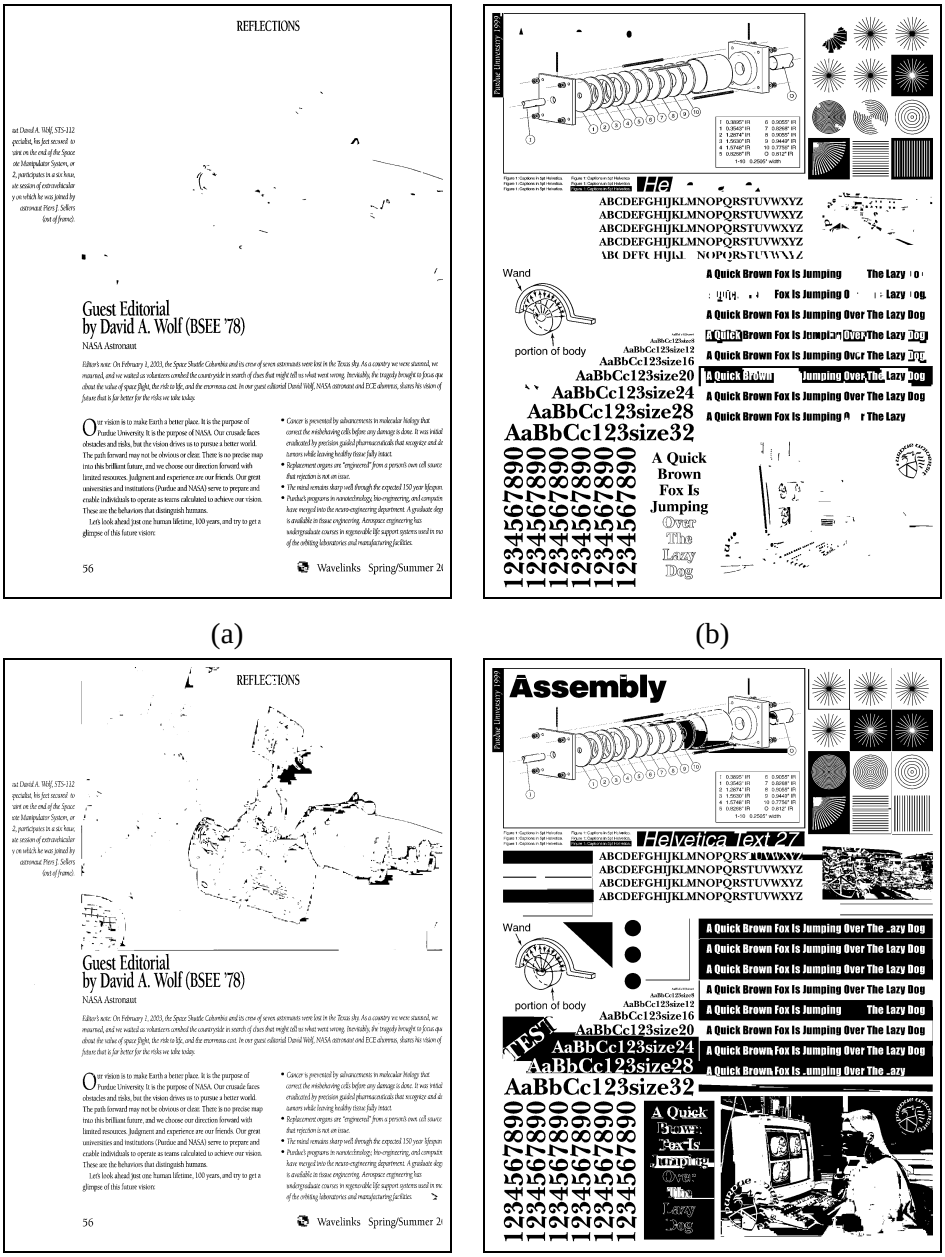
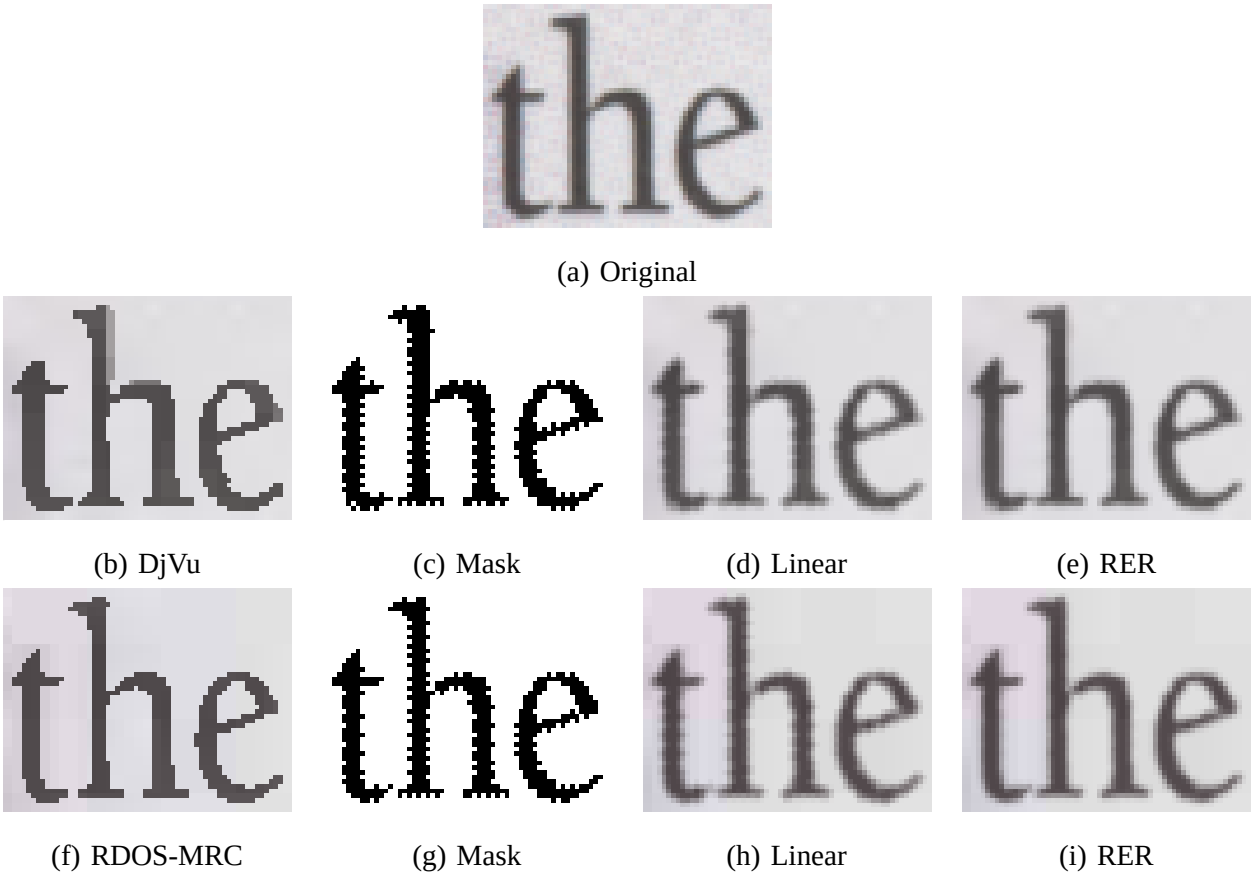


Fig. 15. Binary masks (400 dpi) generated from DjVu and RDOS-MRC. (a) DjVu mask of test image I. (b) DjVu mask of test image II. (c) RDOS-MRC mask of test image I. (d) RDOS-MRC mask of test image II.

over a wide range of bit rates higher than its sharp knee point. In this bit rate range, the MSE distortion is reduced by $55\% \sim 60\%$ by using RER for DjVu and by $45\% \sim 50\%$ by using RER for RDOS-MRC. In addition at each quality level, the RER encoding only increases the bit rate by less than 0.02 bpp, relative to the standard DjVu and RDOS-MRC. This cost is due to the additional dithered edge information encoded in the binary mask. As the bit rate decreases, the RDOS-MRC and RER enhanced RDOS-MRC



(a) Original (b) DjVu (c) Mask (d) Linear (e) RER

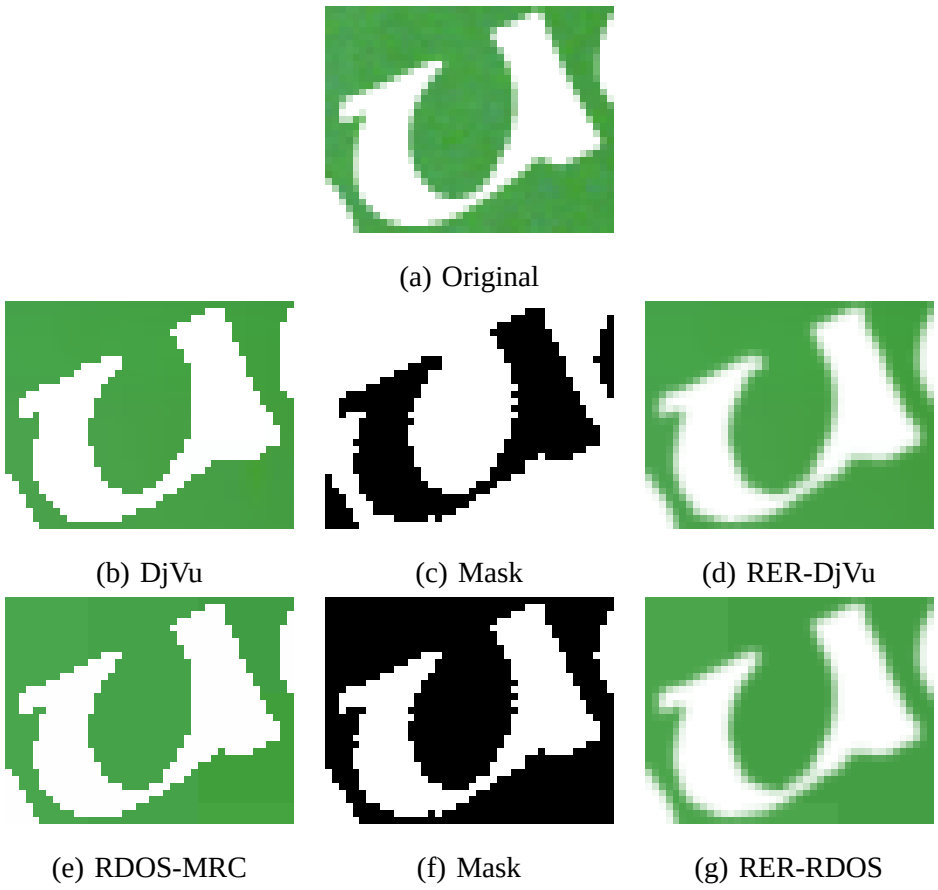
(f) RDOS-MRC (g) Mask (h) Linear (i) RER

Fig. 16. A portion of 400 dpi test image I showing: (a) Original; (b) DjVu at 0.1580 bpp (152:1 compression ratio); (c) Binary mask for RER enhanced DjVu; (d) Linear prediction enhanced DjVu at 0.1557 bpp (154:1 compression ratio); (e) RER enhanced DjVu at 0.1557 bpp (154:1 compression ratio); (f) RDOS-MRC at 0.1582 bpp (152:1 compression ratio); (g) Binary mask for RER enhanced RDOS-MRC; (h) Linear prediction enhanced RDOS-MRC at 0.1524 bpp (157:1 compression ratio); (i) RER enhanced RDOS-MRC at 0.1524 bpp (157:1 compression ratio).

performance become close due to the increasing JPEG compression errors, while the RER enhanced DjVu still preserves signifi cant improvement over the standard DjVu. This is because the wavelet decoder in DjVu can produce much more accurate foreground and background layers than JPEG at low bit rates. Fig. 18(b) shows similar performance for the synthetic test image II.

D. Results of IRER

In this experiment, the 300 dpi document was taken as the input of the encoder, which was formed by decimation of the 600 dpi original document using 2×2 averaging. All output documents of DjVu, RER enhanced DjVu, RDOS-MRC and RER enhanced RDOS-MRC were produced by pixel replication of the 300 dpi decoded documents. For all RDOS-MRC based methods, both background and foreground layers were encoded at 300 dpi (i.e. the same resolution as the mask), and a fi xed segmentation was used



(a) Original

(b) DjVu (c) Mask (d) RER-DjVu

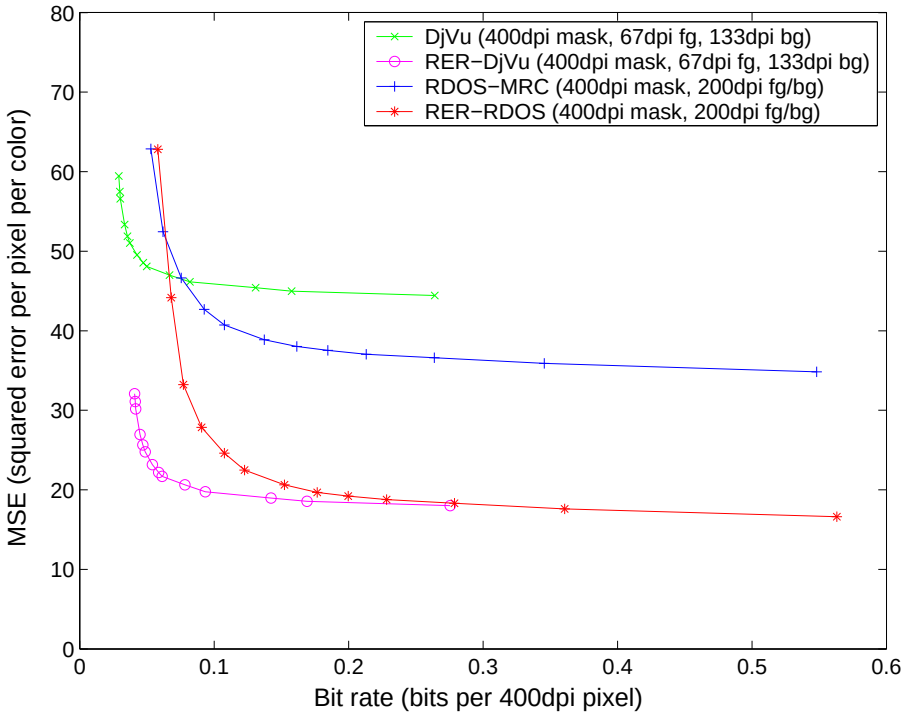
(e) RDOS-MRC (f) Mask (g) RER-RDOS

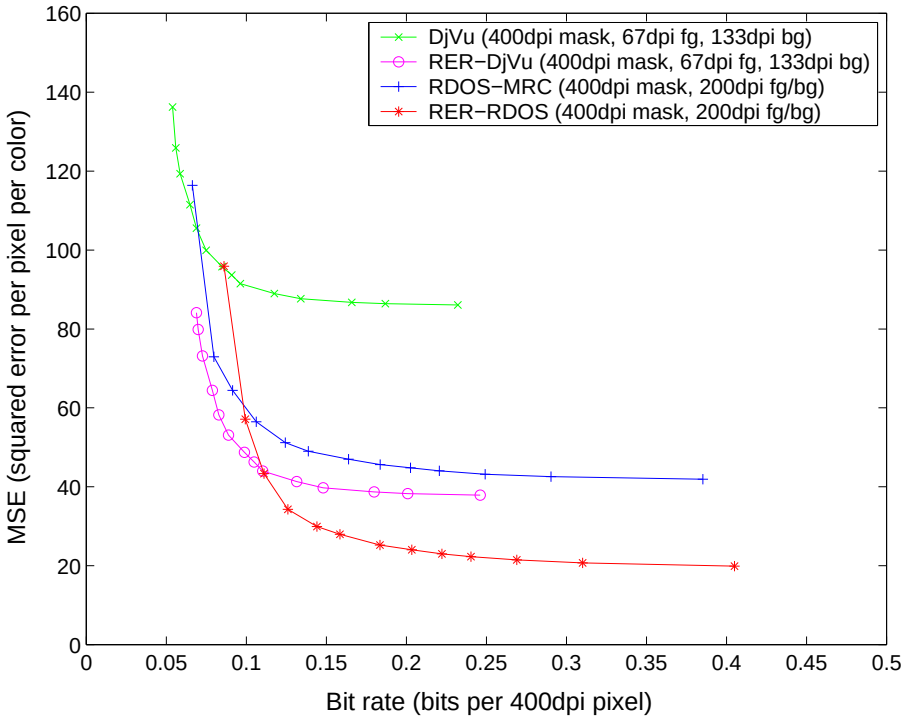
Fig. 17. A portion of 400 dpi test image II showing: (a) Original; (b) DjVu at 0.1437 bpp (167:1 compression ratio); (c) Binary mask for RER enhanced DjVu; (d) RER enhanced DjVu at 0.1399 bpp (172:1 compression ratio); (e) RDOS-MRC at 0.1442 bpp (166:1 compression ratio); (f) Binary mask for RER enhanced RDOS-MRC; (g) RER enhanced RDOS-MRC at 0.1414 bpp (170:1 compression ratio).

for all quality levels. For all DjVu based methods, the foreground and background were encoded at 50 dpi and 100 dpi, respectively.

Fig. 19 shows the comparison among the original 600 dpi scanned image, the images decoded by DjVu, RER enhanced DjVu, IRER enhanced DjVu, RDOS-MRC, RER enhanced RDOS-MRC, and IRER enhanced RDOS-MRC. Notice that the IRER results in both Fig. 19(d) and Fig. 19(g) demonstrate much more accurate renditions of the original 600 dpi document. Comparatively, the results in Fig. 19(b) and Fig. 19(e) all exhibit similar objectionable 300 dpi “jaggie” artifacts around the text edges, where some character shapes are noticeably distorted. Fig. 19(c) and Fig. 19(f) show that the RER method reduces these artifacts but is still limited to 300 dpi output, consequently yielding blurring artifacts.

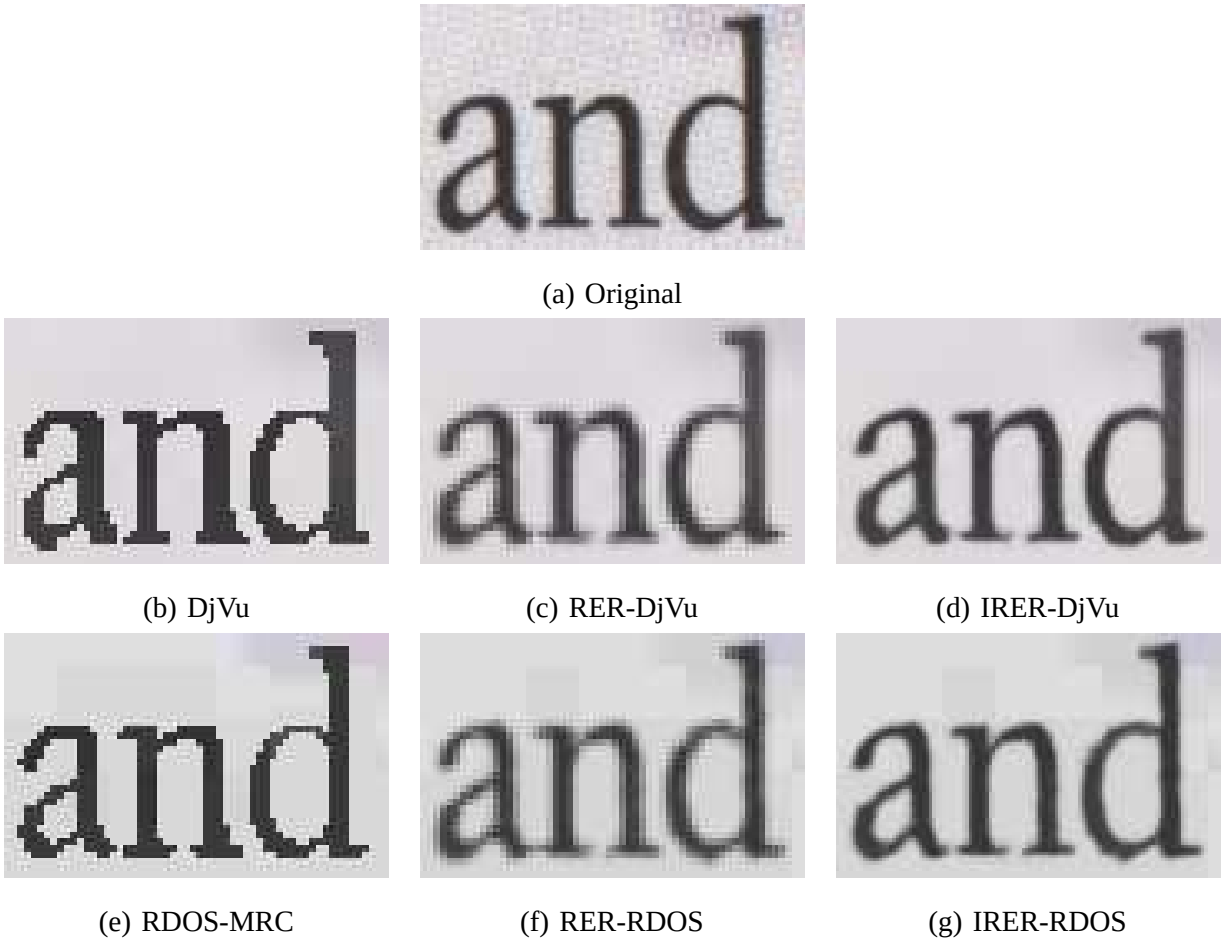
Fig. 20 shows the comparison among the original 600 dpi synthetic image, the images decoded by DjVu, RER enhanced DjVu, IRER enhanced DjVu, RDOS-MRC, RER enhanced RDOS-MRC, and IRER





(b)

Fig. 18. Rate-Distortion performance of DjVu, RER enhanced DjVu, RDOS-MRC and RER enhanced RDOS-MRC on: (a) 400 dpi test image I; (b) 400 dpi test image II.



(a) Original

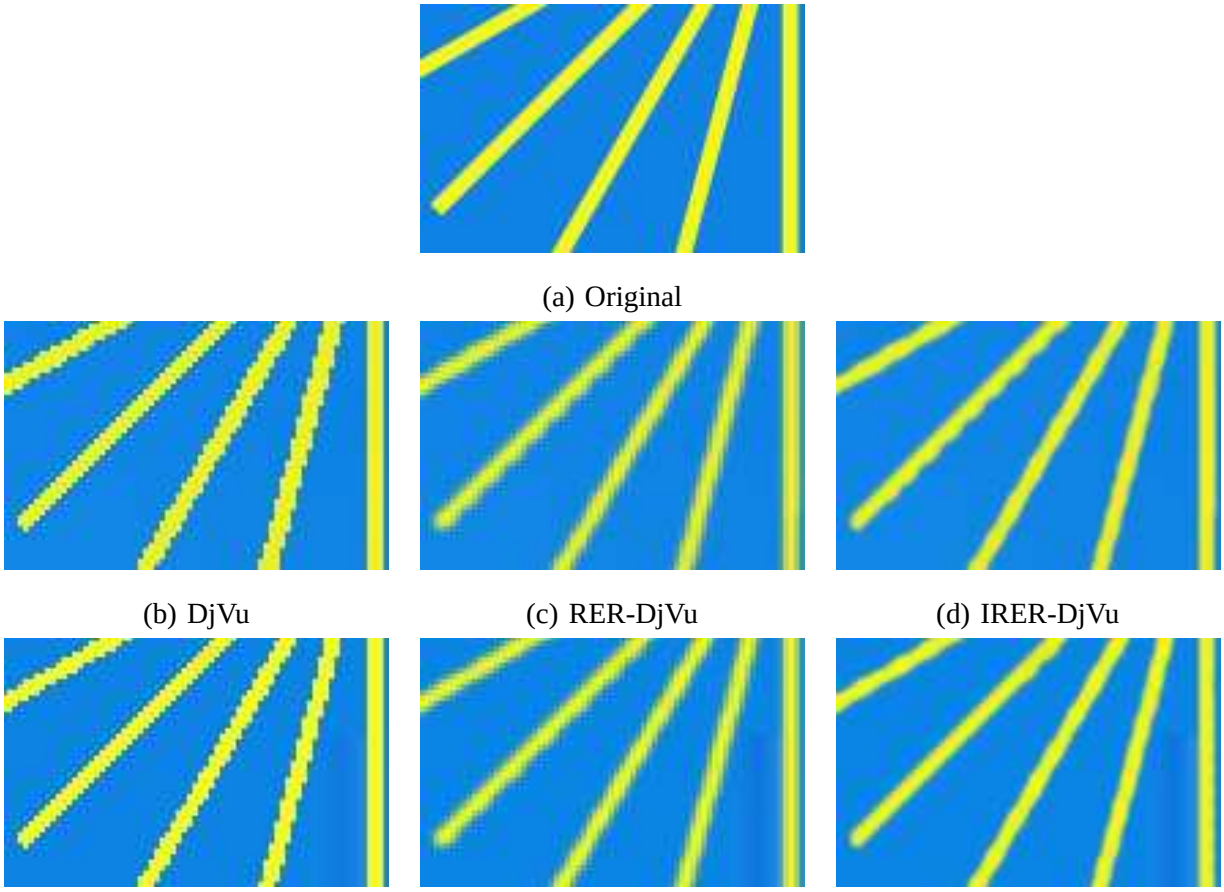
(b) DjVu (c) RER-DjVu (d) IRER-DjVu

(e) RDOS-MRC (f) RER-RDOS (g) IRER-RDOS

Fig. 19. A Portion of 600 dpi test image I showing: (a) Original; (b) DjVu at 0.0561 bits per 600 pixel (428:1 compression ratio); (c) RER enhanced DjVu at 0.0509 bits per 600 pixel (472:1 compression ratio); (d) IRER enhanced DjVu at 0.0508 bits per 600 pixel (472:1 compression ratio); (e) RDOS-MRC at 0.0578 bits per 600 pixel (415:1 compression ratio); (f) RER enhanced RDOS-MRC at 0.0547 bits per 600 pixel (439:1 compression ratio); (g) IRER enhanced RDOS-MRC at 0.0546 bits per 600 pixel (440:1 compression ratio).

enhanced RDOS-MRC. Fig. 20(d) and Fig. 20(g) show the robust adaptivity of the IRER rendition on edges with different orientations. The line edge details are largely recovered and the line straightness and sharpness are well preserved. Notice that in Fig. 20(d), a line is not consistently rendered on both sides, with one side of the edge being more jagged than the other. This problem could be potentially improved through the use of larger window, better training and/or more classes for the nonlinear interpolator.

Fig. 21 illustrates the comparison of rate-distortion performance among DjVu, RDOS-MRC, RER enhanced DjVu and RDOS-MRC, IRER enhanced DjVu and RDOS-MRC. These curves were computed in a manner similar to Fig. 18. The distortion of each 600 dpi decoded document is measured using squared error per pixel per color; and the bit rate is measured in bits per 600 dpi output pixel. The rate-



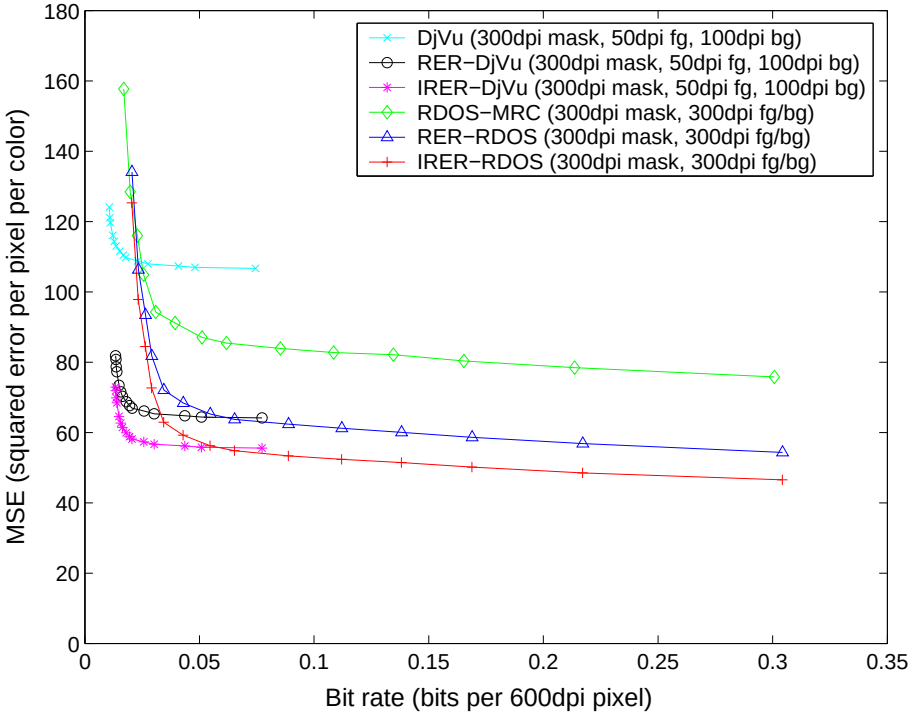
(a) Original

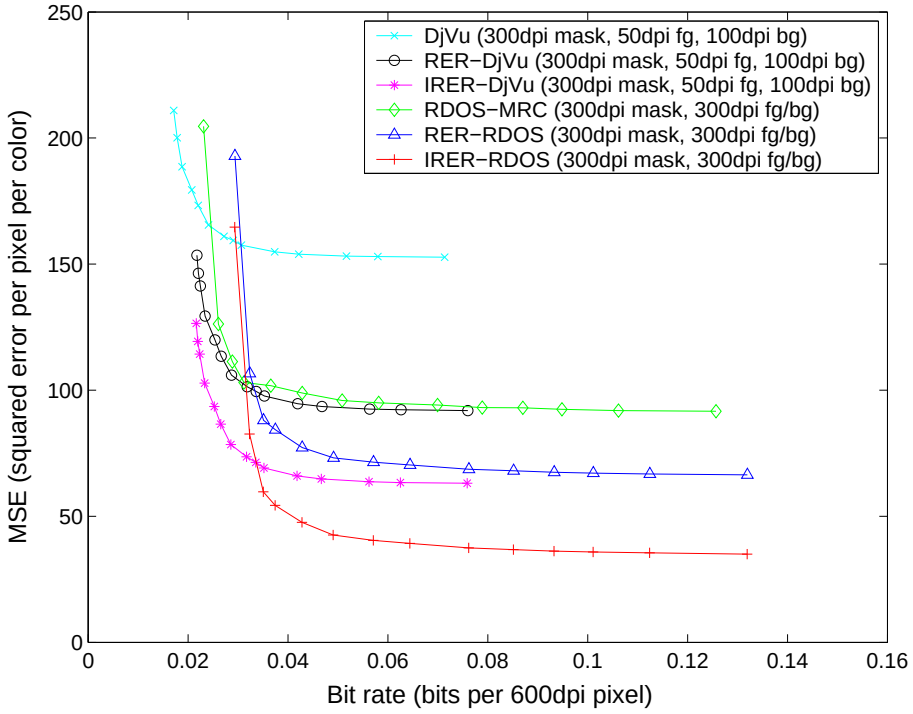
(b) DjVu (c) RER-DjVu (d) IRER-DjVu

(e) RDOS-MRC (f) RER-RDOS (g) IRER-RDOS

Fig. 20. A Portion of 600 dpi test image II showing: (a) Original; (b) DjVu at 0.0569 bits per 600 pixel (422:1 compression ratio); (c) RER enhanced DjVu at 0.0564 bits per 600 pixel (426:1 compression ratio); (d) IRER enhanced DjVu at 0.0562 bits per 600 pixel (427:1 compression ratio); (e) RDOS-MRC at 0.0567 bits per 600 pixel (423:1 compression ratio); (f) RER enhanced RDOS-MRC at 0.0558 bits per 600 pixel (430:1 compression ratio); (g) IRER enhanced RDOS-MRC at 0.0557 bits per 600 pixel (431:1 compression ratio).

distortion curves show the substantial improvement achieved by the RER and IRER methods particularly for the case of synthetic documents. Moreover, the IRER method produces better R-D performance than the RER method over the entire range of bit rate. In particular, the IRER method produces low distortion encodings of the 600 dpi document, but with very high compression ratios on the order of 600:1. These high compression ratios result from the fact that much of the document (corresponding to background and image content) is actually rendered at 300 dpi, and the text and graphic content is rendered at the 600 dpi resolution necessary to preserve its readability, but using only a 300 dpi decoded binary mask. Notice that RER and IRER produce greater reduction in MSE distortion for the synthetic test image II than for the scanned test image I. This is related to the fact that the text and edges in the synthetic





(b)

Fig. 21. Rate-Distortion performance of DjVu, RER enhanced DjVu, IRER enhanced DjVu, RDOS-MRC, RER enhanced RDOS-MRC and IRER enhanced RDOS-MRC on: (a) 600 dpi test image I; (b) 600 dpi test image II.

document are much better shaped with less noise than the scanned document.

VI. Conclusions

In this paper, we have proposed the resolution enhanced rendering (RER) method for implementation with standard mixed raster content (MRC) encoders and decoders. The RER method works by encoding edge detail into the binary mask layer of the MRC document using an adaptive error diffusion method. It then decodes the MRC document by using a nonlinear predictor to determine the relative amount of foreground and background color to apply to each pixel. We propose a method for jointly optimizing the parameters of the RER encoder and decoder to yield minimal document image distortion. The RER method can be extended by incorporating interpolation into the document decoding process, which we refer to as interpolative resolution enhanced rendering (IRER). Our experimental results indicate that the RER method can substantially reduce document image distortion at a fixed bit rate, and the IRER method can achieve high compression ratio (order of 600:1 for 600 dpi document), while retaining good text fidelity of high resolution. In addition, the RER method has the advantage that it can be efficiently implemented in standard MRC encoders and decoders, and it is fully compatible with the existing MRC standard.

Appendix

RDOS-MRC is a 3-layer MRC compression algorithm that is based on a rate-distortion optimized segmentation (RDOS) model[13], [31], [15]. The foreground layer and background layer are compressed at low resolution (typically half of the original) using baseline JPEG with different quality levels, while the binary mask layer is encoded using a lossless JBIG2 encoder. The simple average is used for the decimation of the foreground and background layers. The rate-distortion optimized segmentation (RDOS) algorithm classifies each 8×8 block of pixels into one of four classes: “two-color block”, “two-color inverse block”, “foreground block” and “background block”. Each two-color block or two-color inverse block is represented by a dark color and a light color, each as either foreground layer or background layer, and an 8×8 binary mask layer to identify each pixel between the two colors. For each foreground block, all the pixels belong to the foreground layer, while the background layer is filled in with the mean color of the background layer in the previous block, and the binary mask layer is set to 0. The similar representation is used for each background block. The class of each block is chosen to maximize the rate-distortion performance over the entire document by using the dynamic programming technique.

In our current implementation of RDOS-MRC, a major change has been made on the previous version that used the algorithm in [13]. An extra cost is added to the rate-distortion cost function for each class of

each block in terms of both the horizontal and vertical constraints. Specifically, the extra cost of current block is proportional to the number of pixels in the mask that have different values (i.e. 0 and 1) along the boundary between the current block and its left block or upper block. This encourages the consistency of layer colors between adjacent blocks, thereby producing better quality of binary mask. The examples in Fig. 15 indicate that the current RDOS-MRC segmentation is more efficient and robust than the previous result as shown in [13].

REFERENCES

[1] "JPEG 2000 Image Coding System," *ITU-T Recommendation T.800, ISO/IEC FCD 15444-1*, 2000.

[2] D. S. Taubman and M. W. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Norwell, MA: Kluwer, 2002.

[3] J. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. on Signal Processing*, vol. 41, pp. 3445–3462, December 1993.

[4] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243–250, June 1996.

[5] "Mixed Raster Content (MRC)," *ITU-T Recommendation T.44, Study Group-8 Contribution*, 1998.

[6] R. L. de Queiroz, R. Buckley, and M. Xu, "Mixed Raster Content (MRC) model for compound image compression," in *Proc. of SPIE Conf. on Visual Communications and Image Processing*, vol. 3653, San Jose, CA, January 25-27 1999, pp. 1106–1117.

[7] R. Buckley, D. Venable, and L. McIntyre, "New developments in color facsimile and internet fax," in *Proc. of IS&T/SID Fifth Color Imaging Conference: Color Science, Systems, and Applications*, vol. 3, Scottsdale, AZ, November 17-20 1997, pp. 296–300.

[8] "JPEG 2000 Image Coding System: Compound Image File Format," *ITU-T Recommendation T.805, ISO/IEC FCD 15444-6*, 2001.

[9] 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.

[10] D. Huttenlocher, P. Felzenszwalb, and W. Rucklidge, "Digipaper: a versatile color document image representation," in *Proc. of IEEE Int'l Conf. on Image Proc.*, vol. 1, Kobe, Japan, October 25-28 1999, pp. 219–223.

[11] R. L. de Queiroz, Z. Fan, and T. D. Tran, "Optimizing block-thresholding segmentation for multilayer compression of compound images," *IEEE Trans. on Image Processing*, vol. 9, no. 9, pp. 1461–1471, September 2000.

[12] D. Mukherjee, C. Chrysafis, and A. Said, "JPEG2000-matched MRC compression of compound documents image processing," in *Proc. of IEEE Int'l Conf. on Image Proc.*, vol. 3, Rochester, NY, September 22-25 2002, pp. 73–76.

[13] H. Cheng, G. Feng, and C. A. Bouman, "Rate-distortion based segmentation for MRC compression," in *Proc. of SPIE/IS&T Conf. on Color Imaging: Device-Independent Color, Color Hardcopy, and Applications VII*, vol. 4663, San Jose, CA, January 22-25 2002, pp. 86–97.

[14] Z. Fan and M. Xu, "A simple segmentation algorithm for Mixed Raster Contents image representation," in *Proc. of SPIE/IS&T Conf. on Color Imaging: Device-Independent Color, Color Hardcopy, and Applications VII*, vol. 4663, San Jose, CA, January 22-25 2002, pp. 63–71.

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

[16] 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.

[17] G. Feng, H. Cheng, and C. Bouman, "High quality MRC document coding," in *Proc. of the Image Processing, Image Quality, Image Capture Systems Conference*, Montreal, Canada, April 21-25 2001, pp. 320–325.

[18] C. B. Atkins, C. A. Bouman, and J. P. Allebach, "Tree-based resolution synthesis," in *Proc. of the Image Processing, Image Quality, Image Capture Systems Conference*, Savannah, GA, April 25-28 1999, pp. 405–410.

[19] A. K. Jain, *Fundamentals of Digital Image Processing*. New Jersey: Prentice Hall, 1988.

[20] H. S. Hou and H. C. Andrews, "Cubic splines for image interpolation and digital filtering," *IEEE Trans. on Acoustic Speech and Signal Processing*, vol. 26, no. 6, pp. 508–517, December 1978.

[21] M. Unser, A. Aldroubi, and M. Eden, "Enlargement or reduction of digital images with minimum loss of information," *IEEE Trans. on Image Processing*, vol. 4, no. 3, pp. 247–258, March 1995.

[22] B. Zeng and A. N. Venetsanopoulos, "A comparative study of several nonlinear image interpolation schemes," in *Proc. of SPIE*, vol. 1818, 1992, pp. 21–29.

[23] J. P. Allebach and P. W. Wong, "Edge-directed interpolation," in *Proc. of IEEE Int'l Conf. on Image Proc.*, vol. 3, 1996, pp. 707–710.

[24] K. Jensen and D. Anastassiou, "Subpixel edge localization and the interpolation of still images," *IEEE Trans. on Image Processing*, vol. 4, no. 3, pp. 285–295, March 1995.

[25] S. G. Chang, Z. Cvetkovic, and M. Vetterli, "Resolution enhancement of images using wavelet transform extrema extrapolation," in *Proc. of IEEE Int'l Conf. on Acoust., Speech and Sig. Proc.*, vol. 4, 1995, pp. 2379–2382.

[26] R. R. Schultz and R. L. Stevenson, "A Bayesian approach to image expansion for improved definition," *IEEE Trans. on Image Processing*, vol. 3, no. 3, pp. 233–242, May 1994.

[27] 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.

[28] P. G. Howard, F. Kossentini, B. Martins, S. Forchhammer, and W. J. Rucklidge, "The emerging JBIG2 standard," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 8, no. 7, pp. 838–848, November 1998.

[29] C. B. Atkins, *Classification Based Methods in Optimal Image Interpolation, Ph.D. dissertation*. West Lafayette, IN: Purdue University, 1998.

[30] R. A. Ulichney, "Dithering with blue noise," *Proc. of the IEEE*, vol. 76, pp. 56–79, January 1988.

[31] H. Cheng and C. A. Bouman, "Multilayer document compression algorithm," in *Proc. of IEEE Int'l Conf. on Image Proc.*, vol. 1, Kobe, Japan, October 24-28 1999, pp. 244–248.