15

COMPRESSIONG COMPOUND DOCUMENTS

Ricardo L. de Queiroz
Universidade de Brasília
queiroz@ieee.org

Abstract

In this chapter we introduce the basic methods for the compression of documents in raster format that may contain a mixture of text, graphics and pictures. These documents with mixed contents are called compound documents. We present in detail the multilayer approach to decompose a compound image into homogeneous individual planes, i.e. into images with only text, or pictures, or graphics, etc. We focus on the mixed raster content (MRC) standard as the main compound image representation. The MRC framework facilitates compression and is also part of the JPEG 2000 Part 6 standard (JPM). We describe common imaging models and explain segmentation strategies and goals. A block-based optimized segmentation algorithm is presented, along with its fast approximation. We also describe efficient plane filling algorithms, which are necessary parts of an MRC representation. JPEG 2000’s JPM profile is discussed as a framework for general multipage multilayer document compression. Results are shown along with decompressed images to illustrate the performance of the compound document compression algorithms discussed in this chapter.

15.1 Introduction

Electronic documents are commonplace. From PDF files [1] to fax transmissions, including internal raster representations, a number of electronic formats are used to convey text and pictorial information. Documents are present in a wide spectrum of printing systems and are basically represented in vectorial or raster forms. A document in a vector form is composed by a series of primitives, or instructions such as “render this text at that position.” It often comes as a pair of object and instruction, e.g. the text is the object and the instruction was to render it at a particular position. Same happens to graphics. The document is imaged by “drawing” all “vectors” and applying instructions to image a number of objects. Examples are PostScript [2] and PDF files [1], printer languages
such as PCL [3], graphics primitives in operating systems and even ancient presentation protocols such as NAPLPS [4]. In contrast, raster documents are composed by images of the information to be rendered, e.g. a compressed image or a scan of a magazine page. The advantage of raster images is that they are ready to image, as opposed to vector formats that need to be rasterized first.

It is not much of a challenge to compress vectorized documents since each object can be compressed individually and the whole file can be further compressed losslessly. The real challenge is to compress rasterized compound documents. Compound documents are assumed here as images which contain a mix of textual, graphical, or pictorial contents. A single compression algorithm that simultaneously meets the requirements for both text and image compression has been elusive. As a rule, compression algorithms are developed with a particular image type, characteristic, and application in mind and no single algorithm is best across all image types or applications. When compressing text, it is important to preserve the edges and shapes of characters accurately to facilitate reading. The human visual system, however, works differently for typical continuous-tone images, better masking high frequency errors. Roughly speaking, text requires few bits per pixel but many pixels per inch, while pictures require many bits per pixels but fewer pixels per inch. Document compression is frequently linked to facsimile systems, in which large document bitmaps are compressed before transmission over telephone lines. There is now a focus on new standards to provide color facsimile services over the telephone network and the Internet [5]. When it comes to compound documents different compression algorithms may be applied to each of the regions of the document. This can be accomplished by segmenting the regions or by generating multiple image layers.

15.2 Raster imaging models

A raster bitmap is commonly generated immediately prior to imaging. Hybrid raster images are intermediate formats which can be easily rendered but allow for improved compression and editing. The main problem is that a raster, rather than a vector form, does not distinguish the different objects in a document. If one could separate the objects into different raster images, that somehow could be combined later on, that would facilitate processing the image. There are several imaging models to accomplish the separation of multiple raster by defining a method to recombine (render) multiple raster images into a single bitmap.

15.2.1 Mixed raster content (MRC)

The mixed raster content (MRC) imaging model [6]–[8] enables a multi-layer multi-resolution representation of a compound document, as illustrated in Fig. 15.1. The basic 3-layer MRC model represents a color image as two color-image layers (Foreground or FG and Background or BG) and a binary layer (Mask). The Mask layer describes how to reconstruct the final image from the FG/BG layers, i.e. to use the corresponding pixel from the FG or BG layers when the mask pixel is 1 or 0, respectively, in that position. Thus, the FG layer is essentially poured through the Mask plane onto the BG layer as depicted in Fig. 15.1(a). In reality, the MRC imaging model employs a sequence of image pairs: foreground layers and corresponding mask as illustrated in Fig. 15.1(b). With these, starting from a basic background plane, one can pour onto it a plurality of foreground and mask pairs as in Fig. 15.1(c). Actually, the MRC imaging model allows for one, two, three or more layers. For example, a page consisting of a picture could
The rationale for MRC is that once the original single-resolution image is decomposed into layers, each layer can be processed and compressed using different algorithms. The compression algorithm and processing used for a given layer would be matched to the layer’s content.
15.2.2 Region classification

MRC implies a superposition of frames wherein a selector or mask plane decides from where to render a particular pixel, either from FG or BG planes. However, all the FG and BG layers must be encoded somehow and there is redundant data available to the decoder. Alternatively one can select which regions to assign to BG or FG, but unlike MRC, not storing the image information that is not to be imaged. In this case only one plane of data is made available to the decoder. In effect, the encoder prepares a map like the one depicted in Fig. 15.3 which divides the image into several regions, each region labelled as pertaining to a given layer. Typically the image is divided into blocks and each block is classified as, for example, text, picture, graphics or background. Every class is generally assigned to an image compression method. When rendering the final raster, for a given block, one looks at the map such as the one in Fig. 15.3 to find which coder (layer) contains the block information. The block image information is decoded, the block is rendered and we move on to the next block.

Many image compression methods that deal with compound images use this model [9]–[11]. Note that a method is not MRC compatible just because it employs multiple layers. The important distinction is data redundancy. In MRC, the entire plane is represented. One can render it and print it if desired. When there is pure region classification, every image region goes in only one layer, so that layers are not entire images but data streams. For example, the renderer to form an image block will look at the map and retrieve compressed data from either the JBIG or JPEG stream. Region classification avoids redundancy and is more natural for efficient compression. However, it requires modifying compressors to deal with segmented image data. MRC, on the other hand, employs stock compression only.

15.2.3 Other imaging models

DjVu

DjVu (pronounced as *déjà vu*) [12],[13] is a software and technology for exchanging documents over the web. For that the target is very high compression ratios while preserving comfortable readability. In the DjVu model there are a number of objects to be imaged onto the background plane. An object is a contiguous association of “on” pixels, like letters in a text. Each object is associated with a color and position in the page. In other words, one first images the background and then pastes color objects onto it. Note that it departs from the typical MRC model in which the “object” is within the selector.
plane and colors are conveyed as bitmaps in the foreground plane. In the DjVu model the object itself is associated to the color. Note, also, that the DjVu package provides modes for compatibility with MRC. Nevertheless this imaging model is an interesting solution for a mixed representation.

**Soft masks for blending**

In the typical 3-layer MRC model the output pixel \(x(i, j)\) at position \((i, j)\) will be imaged from either the FG, i.e. \(x(i, j) = f(i, j)\), or the BG, i.e. \(x(i, j) = g(i, j)\). The mask layer pixel \(m(i, j)\) conveys that information of which plane to use for rendering at that point. This is a hard decision between planes. However, it is also possible to use continuous mask and blend the FG and BG planes, e.g.

\[
x(i, j) = m(i, j)f(i, j) + (1 - m(i, j))g(i, j) \quad (15.2)
\]

for \(0 \leq m(i, j) \leq 1\), i.e. the mask is a continuous from 0 to 1 that controls the blending of the FG and BG planes. In the traditional hard decision case, there is no blending and \(m(i, j)\) is either 0 or 1 but nothing in between. The continuous mask option allows for much more flexibility.

For multiple layers the initial canvas pixels \(x_0(i, j)\) are the background pixels \(f_0(i, j)\) and

\[
x_k(i, j) = m_k(i, j)f_k(i, j) + (1 - m_k(i, j))x_{k-1}(i, j) \quad (15.3)
\]

for \(1 \leq k \leq N\), where the mask pixels \(m_k(i, j)\) are continuous from 0 to 1. The final image is \(x_N(i, j)\).

**Residual additive planes**

Additional to selecting or blending planes, one can also use additive planes to correct for errors in a linear way. For example, in the check compression standard [14], there is provided a MRC model plus an additive plane, i.e.

\[
x(i, j) = m(i, j)f(i, j) + (1 - m(i, j))g(i, j) + r(i, j) \quad (15.4)
\]

where \(r(i, j)\) is the residual plane and \(m(i, j)\) can be either binary (as is MRC and the check compression case) or continuous.
15.3 MRC for compression

MRC is the international standard for compound image compression. It was originally approved for use in Group 3 color fax and is described in ITU-T Recommendation T.44 [6]. For the storage, archiving and general interchange of MRC-encoded image data, the TIFF-FX file format has been proposed [8]. TIFF-FX (TIFF for Fax eXtended) represents the coded data generated by the suite of ITU recommendations for facsimile, including single-compression methods MH, MR, MMR [15],[16], JBIG [17], JBIG2 [18] and JPEG [19], JPEG 2000 [20], [21], as well as MRC. As IETF RFC 2301, TIFF-FX [8] is an Internet Standard and it is also the document compression framework for JPEG-2000 Part 6 [22],[23],[24]. MRC has been used in products such as Digipaper [25], DjVu [12] and LuraDocument [26].

As we have discussed, the MRC model can potentially use a number of mask plus foreground image pairs. Nevertheless, unless otherwise noted, we assume a basic 3-layer MRC model, which the reader can easily extend to encompass multiple layers. Once the original single-resolution image is decomposed into layers, each layer can be processed and compressed using different algorithms as shown in Fig. 15.5. The image processing operations can include a resolution change or color mapping. The compression algorithm and resolution used for a given layer would be matched to the layer’s content, allowing for improved compression while reducing distortion visibility. The compressed layers are then packaged in a format, such as TIFF-FX [8] or as a JPM data stream [22],[23] for delivery to the decoder. At the decoder, each plane is retrieved, decompressed, processed (which might include scaling) and the image is finally rendered using the MRC imaging model.

The reason why a model such as MRC works is because one can use processing and compressors tailored to each planes statistics. Thus one can attain improved performance compressing each plane. Hopefully, these gains are enough to offset the expenses in representing redundant data (remember that every pixel position is represented in each of the FG-BG-mask layers). The potential gain of the MRC model for compression can be analyzed under the light of its rate-distortion (RD) characteristics [7]. If the image in Fig. 15.6 is compressed with a generic coder with fixed parameters except for a compression parameter, it will operate under a given RD curve. Another coder under the same circumstances is said to outperform the first coder if its RD curve is shifted to the left (down), i.e. has lower rate for a given distortion or less distortion for a given rate. The rationale for MRC is to split the image into multiple planes as shown in Fig. 15.6 and to apply to each plane a coder (A, B, and C) whose RD curves are better than those...
of the single plane coder. In this case, there is a possibility that the equivalent coder will have better RD curves than the single plane coder, despite the overhead associated with a multi-plane representation.

15.3.1 Object segmentation versus region classification

The degrees of freedom in MRC-based compression are the layer decomposition process (the Segmenter module in Fig. 15.5) and the compressors with their associated parameters for each plane. The encoder and decoder would agree a priori on the compressors, which would be part of the standard employing MRC as an architectural framework. Decomposition affects the operation of the encoder, but not that of the decoder. There are two main approaches to decompose compound images which are illustrated in Fig. 15.7. They are based on object segmentation or on region classification. The image in Fig. 15.7(a) is associated with a BG plane in Fig. 15.7(b) which may contain the paper background along with the continuous-tone picture. The object decomposition relies on identifying text and graphics objects. The concept is that the text or graphics ink is poured through the mask plane onto the BG plane. For this, the mask should
Figure 15.7: Two example segmentation strategies that yield the same image. (a) Original image containing a picture and colored text; (b) BG plane with the canvas and the picture; (c) mask plane containing the text shapes; (d) text associated colors are present at the FG plane; (e) another mask that simply marks the text areas; (f) the correspondent FG plane containing the colored text.

have the contours of text elements, as illustrated in Fig. 15.7(c), while the FG plane in Fig. 15.7(d) contains solid colors of the objects. Thus, the mask image layer would contain text characters, line art and filled regions, while the foreground layer contains the colors of the shapes in the mask layer, i.e. the color of text letters and graphics.

In region classification, regions containing text and graphics are identified and represented in a separate (foreground) plane. The whole region is represented in the foreground plane including the spaces in between letters and such. The mask is very uniform with large patches indicating the text and graphics regions, while the BG plane contains the remaining regions, i.e. the document background itself, complex graphics and/or continuous tone pictures. In Fig. 15.7(e) the mask is actually made of large areas indicating where the text lies. The text itself is contained within the FG plane in Fig. 15.7(f). In the MRC model, which contains redundant image representation, if compression ratio is the main motivation, it is often more useful to employ object segmentation.
15.3.2 Redundant data and segmentation analysis

In an MRC model, image data is redundant. Even if the mask indicates that a particular image position is to be represented for example using the FG data, there is some image data into the BG plane for that position. That information is redundant and does not affect image reconstruction. We will later discuss efficient means to replace redundant data with something that is more easily compressible. This is referred here as plane filling, i.e. we plug holes into the data planes. Note that once the plane filling algorithms are selected, the only degree of freedom in an MRC decomposition is the segmentation of the input compound image into a binary mask. As illustrated in Fig. 15.8 the segmenter finds a suitable mask from the compound image data. Using the mask layer and the input image, the FG and BG layers are found in a deterministic way.

If the mask layer tell the decoder that some pixels are to be imaged from the BG plane, the corresponding pixels from the FG plane are redundant. In effect they are a waste of information which, nevertheless, has to be encoded. The opposite is true for the FG plane. Where the mask indicates the pixels to be imaged from the FG plane, the corresponding BG pixels are redundant too. Actually any value we use for the redundant image region will be irrelevant for reconstruction. In Fig. 15.8 the image block is analyzed by the segmenter which generates a mask with both colors, i.e. assigns pixels to both BG and FG planes. The data filling algorithms will identify where the redundant regions are and replace their pixel values with some “smooth” data in a process that will be explained later. In each plane there are useful regions (labeled $U$) and redundant or “don’t-care” regions (labeled $X$).

In light of the above discussion, we refer the reader to Fig. 15.9 wherein a mask plane is illustrated on the left. For simplicity we used an example of a simple oval shape, but it could be anything, and is typically composed of a number of distinct objects. Let the “white” region be labeled $A$ and the “black” region be labeled $B$. We can also define the transition region $T$ which encompasses a neighborhood near the border between $A$ and $B$ regions. This transition region crosses over the real border between regions and can be further subdivided into regions belonging to $A$ ($T_A$) and $B$ ($T_B$). In this case,
the whole image \( I \) is made of subregions, i.e. \( I = A \cup B \cup T_A \cup T_B \). If the \( A \) region means that the final pixels will be imaged from the FG plane, while \( B \) means one will use the BG plane, then the \( A \) region will mean a useful (\( U \)) region for the FG plane but a “don’t-care (\( X \)) region for the BG plane, and vice versa: region \( B \) means \( U \) for BG and \( X \) for FG. The scheme is illustrated in Fig. 15.9, where either BG or FG is made of \( U \cup X \cup T_U \cup T_X \).

Let us analyse the segmentation in a rate-distortion viewpoint [7]. If the original image is encoded using a single coder \( S \) which does not use MRC, \( R_S \) bits will be spent yielding a reconstruction distortion \( D_S \) such that:

\[
R_S = R_A^S + R_B^S + R_T^A + R_T^B \quad \text{and} \quad D_S = D_A^S + D_B^S + D_T^A + D_T^B, \quad (15.5)
\]

where the distortion model was chosen to be linear, i.e. overall distortion is the sum of local distortions, while \( R_{A,B,T} \) and \( R_{A,B,T} \) are the rate and distortions for each of the plane regions (see Fig. 15.9). If the image is split into the 3 planes (FG, BG, Mask) corresponding to the MRC model, then the overall rate and distortion are given by

\[
R = R_M + \sum_{\Psi = FG,BG} \sum_{\Omega = A,B,T_A,T_B} R^\Omega_{\Psi} \quad \text{and} \quad D = \sum_{\Psi = FG,BG} \sum_{\Omega = A,B,T_A,T_B} D^\Omega_{\Psi}, \quad (15.6)
\]

Note that the mask is encoded without distortion and that \( X \) pixels, i.e. region \( B \) in the FG plane and region \( A \) in the BG plane, do not contribute to overall distortion. Thus

\[
D = D_{BG}^B + D_{BG}^T + D_{FG}^A + D_{FG}^T. \quad (15.7)
\]

If one wants the MRC scheme to outperform the single coder, it is necessary that either or both \( R < R_S \) and \( D < D_S \). It is sufficient to have

\[
R < R_S \quad \text{and} \quad D < D_S. \quad (15.8)
\]
In a simple coding scenario where the coder for the FG and BG planes is the same as the single coder, we can make the following assumptions: \( R_{BG} = R_{B} \), \( R_{FG} = R_{S} \), \( D_{BG} = D_{B} \), \( D_{FG} = D_{S} \). Furthermore, in general transform coding, the transform bases will likely extend across the region boundaries (that is why we made the transition regions in the first place) so that it is unlikely that one can separate the rate produced by either \( T_A \) or \( T_B \). Hence we define \( R_T = R_{TA} + R_{TB} \), \( R_{FG} = R_{FA} + R_{FB} \), and \( R_{BG} = R_{BA} + R_{BB} \), so that

\[
D_{S} - D = D_{S}^{TA} + D_{S}^{TB} - D_{S}^{TA} + D_{S}^{TB},
\]

which is the overhead rate, due to the mask and to redundant data in the continuous planes. Reduction in rate and distortion are achieved iff

\[
D_{TA} + D_{TB} < D_{S}^{TA} + D_{S}^{TB}
\]

So, following the analysis of this simple example, we see that transition regions are the main regions where compression can be improved by using MRC. In more detail, improvement comes when: (a) distortion in the transition region is less than in the single coder; (b) the savings in encoding the transition regions (in both BG and FG) planes compared to the single coder are enough to offset the expenditure of bits to encode the overhead.

With text object segmentation, (15.11) and (15.12) are usually satisfied. In general, \( R_{BG} < R_{S} \) and \( R_{FG} < R_{S} \). However the decomposition has to be done in such a way that there will be enough transitions in the image to allow enough savings. Furthermore, the regions chosen to be transitions have to be such that they lead to large savings in bit rate in each plane in order to compensate for the redundant information and the overhead.

In an MRC approach, the plane-filling pre-processor (see Fig. 15.8) can very well replace pixels in redundant regions with any computer generated data which would reduce the most of the distortion and bit rate, i.e. to ensure that (15.11) and (15.12) are satisfied. With text segmentation, the transition in the mask occurs for edges in the original image. Hence, \( R_{S} \) and \( D_{S} \) are very high. If, for example, the transition region in each plane is made very smooth, not only will the distortion decrease but the bit-rate can be kept very small. For smooth enough transitions and if we discard the terms \( R_{BG} \) and \( R_{FG} \), then the trade-off of MRC can be summarized as:

\[
R_{S} < R_{S}^{T}.
\]
In reality, a typical coding scenario is usually more favorable to MRC than the above example. This is because coders for foreground and background can be selected to outperform the single coder, while the mask plane often compresses very well. For example if the text is placed into the mask, techniques such as JBIG, MMR, JBIG-2 can compress text well. The FG would contain mainly text color and can be largely compressed. The background plane would contain the pictorial images and the paper texture, which are features that do not contain high-resolution details. In that case, moderate subsampling can be carried before compression. The different nature of the data in each plane allows for very efficient compression with lower error visibility.

15.3.3 Plane filling

In the previous section we assumed that plane filling algorithms can reasonably smooth transitions. Without loss of generality, we can address any plane (FG or BG) individually, by referring to its $X$ and $U$ regions. We want to replace the data in the $X$ region (and $T_X$ in Fig. 15.9) by any data that would improve compression. The overall goal is to reduce both rate and distortion, i.e. to minimize \[ J = R + \lambda D, \] (15.14)

where $\lambda$ controls the rate-distortion trade-off. Assuming rate and distortion are additive per regions (assuming they are independent), we have

\[ J = (R^U + R^X + R^T) + \lambda(D^U + D^X + D^{T_U} + D^{T_X}) \] (15.15)

where the superscript indicates the regions and as we discussed $R^T = R^{T_U} + R^{T_X}$. Note that since a redundant ($X$) region is irrelevant for reconstruction, then $D^X = D^{T_X} = 0$. Also note that since the replacement of redundant data does not affect the $U$ region, the minimization of the above cost function is equivalent to minimizing

\[ J = R^X + R^T + \lambda D^{T_U}. \] (15.16)

i.e. it is equivalent to minimize rate in the $X$ region and to make it RD efficient at transitions.

True optimality is a very ambitious goal. The alternatives are too many to consider: layers can be resized or further processed and there are too many compression options, ranging from the transform type, through the wavelet type, to the choice of quantizers and entropy coders, etc. It is impractical to optimize the redundant data without fixing all these compression parameters, however there is a good compromise with a practical solution which aims to work well across a wide range of applications [27].

In order to minimize (15.16), it seems to be reasonable to apply smooth (flat) data to the redundant region. That would definitely reduce $R^X$ to its virtual minimum. The question is what to do with the transitions, i.e. how to minimize $R^T + \lambda D^T$. If we would do that blindly, the best intuitive solution is to make the transition as smooth as possible. Smooth patterns tend to produce less bits and cause less distortion in most popular image coders.

The problem is to generate smooth transitions. Fig. 15.10 has an illustration of a 1D signal (a) which is masked using the mask in (b) yielding the signal in (c), where the redundant area was replaced by a constant value. If we simply filter the signal in Fig. 15.10(c) we obtain the signal in (d). After applying the mask, assuming we do not touch the useful region, the reconstructed signal is as in Fig. 15.10(e). Note that there
Figure 15.10: 1D example. (a) signal; (b) mask; (c) masked layer; (d) filtered version of (c); (e) masked filtered layer.
still exist a discontinuity, which is caused by the symmetry of the filtering process around the edge. A good solution is to use a segmented filtering [27], where filter weights are exaggerated for pixels in the useful region of the layer. In 1D,

$$y(n) = \frac{\sum_{k=-L}^{L} h(k, n)x(n+k)}{\sum_{k=-L}^{L} h(k, n)}$$  \hspace{1cm} (15.17)$$

where \( h \) is a time varying filter of \( 2L + 1 \) taps. Its weights are dependent on the mask values \( m(n) \) as:

$$h(k, n) = \begin{cases} 1 & \text{if } m(n+k) = 0 \\ Mf(k) + 1 & \text{if } m(n+k) = 1 \end{cases}$$  \hspace{1cm} (15.18)$$

where \( f(k) \) is a filter window such as Hamming to de-emphasize distant pixels within the useful region. The result of applying the segmented filter is shown in Fig. 15.11 for \( L = 16 \), by varying \( M \) and for a rectangular (uniform) and for a Hamming window. Note how the discontinuity is largely decreased. The case \( M = 0 \) is equivalent to a straight averaging filter without any emphasis. This filtering is very easy to implement. For uniform windows, like the averaging filter case, complexity is virtually independent of window size, and very large filters are easily implemented.

This approach assumes a generic coder. If the coder is known, there are ways to further optimize the data filling process. One example is the method used by DjVu in which the wavelet transform is known and a set of interactive projections are made in order to slowly approach a stable and efficient solution.

Other solutions exist for JPEG and operate over small \( 8 \times 8 \) blocks. As in the general method, the pre-processor receives an input block and, by inspecting the binary mask, labels the input block pixels as useful (\( U \)) or “don’t care” (\( X \)).

The spatial domain algorithm is relatively simple and inexpensive [27]. If there are 64 \( X \)-marked pixels, the block is unused and we output a flat block whose pixels are the average of the previous block (because of JPEG’s DC DPCM). If there are no \( X \)-marked pixels, the input block is output untouched. If there is a mix of \( U \)- and \( X \)-marked pixels, we follow a multi-pass algorithm. In each pass, \( X \)-pixels which have at least one \( U \)-pixel as a horizontal or vertical neighbour (i.e. in the \( N_4 \) or NSEW neighbourhood as indicated in Fig. 15.12) are replaced by the average of those neighbours as illustrated in Fig. 15.12. In the next pass, those pixels that were replaced are marked \( U \) for the next pass. Fig. 15.12 illustrates two steps, while the process is continued until there are no
Figure 15.12: Top: an example block mask and respective labels for layer pixels. Bottom: two further stages of the iterative spatial domain block filling algorithm. Shaded pixels are the \( X \) pixels are replaced at each stage by the NSEW neighbor average, thus being relabeled as a \( U \) pixel.

\( X \)-pixels left in the block. The aim of the algorithm is to replace the unused parts of a block with data that will produce a smooth block based on the existing data in the \( U \)-marked pixels. Its disadvantage is that the \( X \)-marked pixels are just influenced by the bordering \( U \) pixels, i.e. an internal \( U \) pixel does not affect data filling. This is acceptable for most applications.

There is a more expensive data filling alternative involving the computation of the DCT [27]. In this method: (i) Initialize \( X \)-pixels in any way, for example as the average of the \( U \)-pixels. (ii) Transform, quantize and inverse transform the block, obtaining a new set of pixels in the block (call them \( X' \) and \( U' \) pixels). (iii) Replace \( X \)-pixels by \( X' \)-pixels in the original block. (iv) Repeat the transformation and replacement process until convergence is reached. Convergence is achieved when \( X' \) and \( X \) pixels are identical or close within a certain prescribed tolerance. It usually happens after very few iterations. An illustration, Fig. 15.13 shows an example block, its respective mask and the resulting block using DCT domain algorithm. For comparison, a block resulting from the spatial domain algorithm is also presented.

### 15.4 A simple MRC: JPEG+MMR+JPEG

A simple scheme that can yield good results utilizes a 3-layer model and simple, standard coders. The most important step in an MRC representation is the segmentation which, along with plane filling, define the MRC plane decomposition. For this simple case, there are efficient segmentation algorithms and block filling methods that enable efficient document compression. There are only the mask, BG and FG planes. The FG plane should contain the text and graphics colors. The BG plane should contain the document background and pictorial data. The mask actually should contain the text and graphics shapes.

The reason to use MRC for compression of compound images is because it is a standard. As a standard it is intended to employ standard compressors as well. JPEG is the most widely adopted compression method for color images and is well suited to compress
Figure 15.13: Iterative DCT-domain block filling. Top: block and mask. Left: 3 steps in the DCT domain algorithm. Right: spatial domain method result.
both the FG and BG layers. For simplicity, one can use MMR to compress the binary plane, although any other binary coder may be applied. Because of the large number of variables involved, we want to keep the MRC model as simple as possible and it does not include resizing the layers before compression.

15.4.1 Computing rate and distortion per block

JPEG is a block-based compression method. In the case of block-based compression we use block-based segmentation as well. The image is divided into blocks of $8 \times 8$ pixels. For the $n$-th input pixel block $\{x_n(i,j)\}$, the segmenter generates a binary mask block $\{m_n(i,j)\}$ with the same dimensions. The data-filling processor generates the layer blocks $\{L_n^{(FG)}(i,j)\}$ and $\{L_n^{(BG)}(i,j)\}$. The FG/BG blocks are compressed and decompressed as $\{\hat{L}_n^{(FG)}(i,j)\}$ and $\{\hat{L}_n^{(BG)}(i,j)\}$, from which the block can be reconstructed as

$$\hat{x}_n(i,j) = m_n(i,j)\hat{L}_n^{(FG)}(i,j) + (1 - m_n(i,j))\hat{L}_n^{(BG)}(i,j).$$  \hfill (15.19)

The distortion for this block is

$$D_n = \sum_{ij} (x_n(i,j) - \hat{x}_n(i,j))^2.$$  \hfill (15.20)

The (estimated) rate for a given block is given as:

$$R_n = R_n^B + R_n^M + R_n^F,$$  \hfill (15.21)

where $R_n^B$, $R_n^M$ and $R_n^F$ are the estimated rates for compressing $\{L_n^{(BG)}(i,j)\}$, $\{m_n(i,j)\}$ and $\{L_n^{(FG)}(i,j)\}$, respectively. The reason for estimating as opposed to computing the actual rates is because of the interdependence among blocks. Even though the FG/BG layers are JPEG compressed, the compression of the mask plane is not block based. Binary coders generally rely on run-lengths, or line-by-line differential positions, or even object properties. Hence, for the Mask layer block, it is very difficult (if not impossible) to accurately determine the amount of bits a single block will generate. Therefore, the contribution of a single block to the overall rate is not direct and one has to estimate the compressed rate for a given mask block. Furthermore, blocks in JPEG are not completely independent since there is a differential encoding of the DC coefficients. Fortunately, the number of bits saved by using block-to-block DC differential encoding in JPEG is not too significant compared to the overall bit rate per block. As for the mask, one solution is to estimate the mask rate by counting the number of horizontal transitions, to which one applies a fixed average penalty (e.g. 7 bits per transition). So, for a block with $N_t$ transitions,

$$R_n^M = N_t \times \text{penalty}.$$  \hfill (15.22)

15.4.2 Optimized thresholding as segmentation

As we discussed, the mask and the input block define the other layers (for a fixed filling algorithms and without spatial scaling). Our goal is to find the best mapping from $\{x(i,j)\}$ to $\{m(i,j)\}$ which will optimize compression in a rate-distortion (RD) sense. We start by breaking the problem (image) into $8 \times 8$-pixel blocks $\{x_n(i,j)\}$ blocks and finding the best mapping $\{x_n(i,j)\}$ to $\{m_n(i,j)\}$. Then, for each input block, there
Dear Paul,
I was delighted to hear from you last week. Fatti and I had a wonderful time during our week-long summer vacation. The weather was excellent, and the food was absolutely exquisite. I hope that we can repeat this next year, and that you will join us too.

We came back with a lot of fantastic memories, which we would like to share with you through some snapshots that we took.

Our favorite is this picture of us aboard the "Top Hat," which I have posted into this letter using some really neat advanced digital imaging technology on my home computer. We will also share the rest of you on a CD-ROM soon. Wishing you the best.

Love,
Susan

Figure 15.14: An example image and its ideal mask.

Figure 15.15: Zoomed portions of the MRC decomposed layers. The fast variance-based block segmentation method was used along with the spatial block-based data-filling algorithm. It is shown the reconstructed image along with the BG-Mask-FG layers.
are $2^{64}$ possible mask blocks. In order to simplify the search algorithm we also impose restrictions on the quantizer table used to JPEG compress each block. Not only we use the same quantizer table $\{q(i,j)\}$ for both FG and BG planes, but we also use a scaled version of JPEG's default table $\{q_d(i,j)\}$ [19] as $q(i,j) = Qq_d(i,j)$. This simplification allows us to control rate and distortion as a function of a single variable $Q$ as opposed to 128 of them. For a given image block $\{x_n(i,j)\}$, using a particular coding quantizer scale $Q$, the RD points for this block, i.e. $R_n$ and $D_n$ will depend on $\{x_n(i,j)\}$, $Q$ and on the mask $\{m_n(i,j)\}$. Here, we want to optimize the segmentation in an RD sense, i.e. to minimize a block cost function

$$J_n = R_n + \lambda D_n,$$  \hspace{1cm} (15.23)

so that $\lambda$ indicates an operating point to control the RD trade-off for the given block. Hence, $\{m_n(i,j)\}$ is a function of $\lambda$, $Q$ and of $\{x_n(i,j)\}$ of course.

We want to find each block mask $m_n(i,j)$. The simplest model for a compound image is based on the histogram of the block pixels. Pictorial, background and text-edge areas should have histograms which are dense, flat, and bimodal, respectively. One simple approach is to find the bimodal blocks and to cluster the pixels around each of its modes. Whatever method is used to perform clustering or test bimodality, the pixels will be divided by some sort of threshold. In block thresholding the mask is found as

$$m_n(i,j) = u(t_n - x_n(i,j) - 1)$$  \hspace{1cm} (15.24)

where $t_n$ is the block’s threshold and $u(k)$ is the discrete step function (equals 1 for $k \geq 0$ and 0 otherwise). In effect, pixels below the threshold are placed in the BG layer. Since there are 64 pixels in a block, there are at most 64 different meaningful threshold values, whereby setting $t_n$ to be less than the darkest pixel forces the Mask block to be uniform, i.e. all samples imaged from one of the layers. A 65th threshold value, $t_n$ greater than the brightest pixel, can be ignored since it achieves the same objective as the first. It has been shown that thresholding yields masks whose performance is among the best possible among all $2^{64}$ $(R_n, D_n)$ pairs [28]. In other words, thresholding is RD-efficient.

The quest is to find the best threshold value $t_n$ in an RD sense, from which one finds the mask using (15.24). In a block there are 64 pixels and therefore only up to 64 threshold values need to be tested. If we sort the block pixels $x_n(i,j)$ into a sequence $p(k)$, for each $t_n = p(k)$, we evaluate

$$J_k = R_n(k) + \lambda D_n(k),$$  \hspace{1cm} (15.25)

where the index $k$ denotes measurements for the $k$-th threshold tested. We recall that $\lambda$ was defined as a control parameter and is used here as a Lagrange multiplier, and that the computation of $R_n$ and $D_n$ assumed a particular $Q$. Both $\lambda$ and $Q$ are fixed for all image blocks. This is so, because it is well known that, for optimality, blocks should operate at the same slope on their RD curves [29], and because baseline JPEG does not allow for changing quantizer tables within an image. We test all $p(k)$ in a block and select the index $k = k_o$ for the minimum $J_k$. Then, $m_n(i,j)$ is found using (15.24) for $t_n = p(k_o)$.

So, we optimize the sequence $\{t_n\}$, block by block, for fixed external variables $\lambda$ and $Q$. Note that the overall $R$ and $D$ are functions of both $\lambda$ and $Q$, i.e. $R(\lambda,Q)$ and $D(\lambda,Q)$. Given a budget $R_b$ (or $D_b$), the goal is to minimize

$$\min_{\lambda,Q} D(\lambda,Q) \bigg|_{R(\lambda,Q) \leq R_b} \quad \text{or} \quad \min_{\lambda,Q} R(\lambda,Q) \bigg|_{D(\lambda,Q) \leq D_b},$$  \hspace{1cm} (15.26)
or, equivalently, we are interested in the lower convex hull (LCH) for a bounded RD region. The search of the 2D space can be very expensive. However, there are simplifying circumstances which may reduce the search. It can be shown that an algorithm that will fulfill (15.26) is as follows [28]:

1. Select a quantizer scale $Q_d$.
2. For every block, input $x_n(i, j)$ and find $t_n$ which minimizes $J = R(t_n, Q_d)$.
3. Obtain mask $m_n(i, j)$ for each block using (15.24).
4. With resulting Mask layer in hand, compress FG/BG layers using another scaling factor $Q_c$.
5. Verify overall rate $R$ (or $D$).
6. If $R$ (or $D$) is not within parameters, adjust $Q_c$ and go to step 4.

Let $C_J$ denote the complexity (in terms of operations) of JPEG compressing a block. It takes about $3C_J$ to test one threshold. If, in average, $k_t$ thresholds need to be tested per block, the segmenter complexity per block for the algorithm is $C = 3k_tC_J$. For predominantly binary images $k_t$ is very small (minimum 2) while for pictures it can be up to 64, i.e. $6C_J \leq C \leq 192C_J$.

### 15.4.3 Fast thresholding

There is, however, a faster and yet efficient technique to obtain $t_n$ [28]. The largest complexity in the RD-optimized algorithm comes from simulating compression of the FG/BG layer blocks in order to compute an estimation of the overall RD point. We want to avoid those computations as well as the block-filling operation. As we discussed, segmentation of bimodal blocks is essentially a method of finding a suitable threshold that would divide the block into two nearly-uniform regions. One can modify the cost function to reflect this property. We also want to add a penalty for mask transitions. For non-bimodal blocks, the cost should be lower for uniform masks than for non-uniform ones. Once the block is thresholded into two sets, measures of variance or entropy should be good estimators of how similar the set members are. We have chosen to use the variance measure not only because the variance is simpler to compute than the entropy, but it also serves as a good rate estimator. Intuitively, if a block has low variance, it should be compressed well with a small distortion.

For each block, we sort its pixels in $p(k)$ just like in the RD case. However we seek to minimize the following cost function

$$J = \alpha_1 V_{BG} + \alpha_2 V_{FG} + \alpha_3 N_t$$  \hspace{1cm} (15.27)

where $\alpha_i$ are weighting factors, $V_{BG}$ and $V_{FG}$ are the variances of pixels in the BG and FG layer blocks. $N_t$ is the number of horizontal transitions of the mask block (the first column of the block uses as reference the last column of the previous mask block, just like in the RD-optimized search case). For a given threshold, a mask block $\{m_n(i, j)\}$ is obtained and we define two sets:

$$X_f \equiv \{ x_n(i, j) \mid m_n(i, j) = 1 \}$$
$$X_b \equiv \{ x_n(i, j) \mid m_n(i, j) = 0 \}.$$  \hspace{1cm} (15.28)
We define \( n_f \) and \( n_b \) as the number of pixels in the set \( X_f \) and \( X_b \), respectively, where obviously \( n_f + n_b = 64 \). Then, variances are computed as:

\[
V_{BG} = \frac{\sum_{ij} x_n(i,j)^2}{n_b} - \left( \frac{\sum_{ij} x_n(i,j)}{n_b} \right)^2,
\]

\[
V_{FG} = \frac{\sum_{ij} x_n(i,j)^2}{n_f} - \left( \frac{\sum_{ij} x_n(i,j)}{n_f} \right)^2
\]

which can be efficiently implemented. Since thresholds are sorted, as we increment \( k \), we will be effectively moving pixels from the set \( X_b \) to the set \( X_f \). Thus, part of the computation does not change in each step. First we set the mask to be all zeros effectively placing the whole block in the background, which is equivalent to set \( t_n \) to be the smallest of \( x_n(i,j) \). Then, we set \( k = 0 \) and initialize the following variables

\[
s_b = \sum_{ij} x_n(i,j) ; \quad v_b = \sum_{ij} x_n^2(i,j) ; \quad n_b = 64,
\]

\[
s_f = v_f = n_f = N_i = 0
\]

We then compute

\[
V_{BG} = \frac{v_b}{n_b} - \left( \frac{s_b}{n_b} \right)^2 \quad V_{FG} = \frac{v_f}{n_f} - \left( \frac{s_f}{n_f} \right)^2
\]

where, if \( n_f \) or \( n_b \) are 0, the corresponding variance is set to 0. Next, we compute (15.27).

As we increase the threshold to the next pixel in the sorted list, we increment \( k \), and the mask changes so that we need to recompute \( N_i \). Some \( n_p \) pixels that form a set \( X_p \) are then moved from \( X_b \) to \( X_f \). Hence, we have to compute \( s_p = \sum_{X_p} x_n(i,j) \) and \( v_p = \sum_{X_p} x_n^2(i,j) \). Then, we update our variables as

\[
s_f = s_f + s_p ; \quad v_f = v_f + v_p ; \quad n_f = n_f + n_p
\]

\[
s_b = s_b - s_p ; \quad v_b = v_b - v_p ; \quad n_b = n_b - n_p
\]

and recompute \( V_{BG}, V_{FG} \) and \( J_k \). We repeat the process until \( X_b \) is empty. We test all 65 possibilities, i.e. from \( X_f \) empty to \( X_b \) empty in order to make the algorithm symmetric. We select \( t_n = p(k) \) for \( \min_k(J_k) \) and we compute the final mask using (15.24).

The overall computation (in terms of operations) is much more reasonable than the RD-optimized counterpart. The overall processing has a computational complexity \( C \) not superior to simply compressing the block with JPEG once, i.e. \( C \approx C_f \).

As for the weights, without loss of generality we can normalize one of them (e.g. \( \alpha_1 = 1 \)). The choice of weights is empirical, however there are some few guidelines that we can use for our advantage. We want to place the complex graphics and pictures in only one of the planes and that can be done with \( \alpha_2 \neq \alpha_1 \). Also, variances are much larger numbers than the number of transitions (given that pixels range from 0 to let us say 255). Thus, \( \alpha_3 \) should be a very large number. In our experiments, the values of \( \alpha_2 = 5 \) and \( \alpha_3 = 200 \) were efficient for segmentation of images containing sharp contrasting black-on-white text. Some masks are shown in Fig. 15.16 for the cases where \( \alpha_1 = \alpha_2 = 1, \alpha_1 = 1 \) and \( \alpha_2 = 5 \), and the case \( \alpha_1 = 5 \) and \( \alpha_2 = 1 \). Although all cases segment well the text, the placement of picture blocks differs. It might be advantageous for other reasons to use \( \alpha_1 = 1, \alpha_2 = 5 \) in order to place the picture along with the background. These values are recommended.
22 Chapter 15

15.4.4 Performance

We can compare the MRC performance using the images shown in Fig. 15.17. The images range from purely graphics (graphics) to purely pictorial (baby), with two other mixed images with graphics (compound1) and pictorial (wine) dominance. Clearly, the more graphics there are, the higher the advantage over a single coder such as JPEG. For an image such as “baby”, our MRC approach has the disadvantage of encoding the overhead of two planes and is expected to be outperformed by JPEG. We compared the following compressors: (i) MRC with optimized thresholding segmentation and $Q_d = 1$; (ii) MRC with a variance-based thresholding and $Q_d = 1$; (iii) single plane JPEG 2000 and (iv) single plane JPEG. For the MRC approach, we computed RD curves by varying $Q_c$, which are shown in Fig. 15.18. The distortion measure chosen was PSNR [29] and the plots present results in differential PSNR compared to JPEG, i.e. how many dB improvement would there be if we replace JPEG by the respective coder (MRC or JPEG 2000).

The PSNR difference against JPEG is extremely large for the graphics case since MRC quickly approaches the lossless state. The image compound1 is one of the best representatives of the target compound images. In this case, the PSNR difference is a staggering 12 dB over JPEG and several dB over JPEG 2000. The performance of the variance-based method is very close to that of the RD-based one, except for pictorial images. As the image becomes purely pictorial, the losses are about or below 1 dB for the RD-based segmentation compared to JPEG. This small loss is a very positive sign: even if by mistake a pictorial image is to be segmented, smart segmentation can minimize losses. Apart from MRC approaches, JPEG 2000 serves as an upper bound in single-layer performance. A comparison between JPEG-2000-based MRC and JPEG 2000 will be carried in the next section.

A sample of reconstructed images for comparison are shown in Fig. 15.19. The superior quality of the MRC-coded image over the JPEG-coded one is easily noticeable in both pictorial and text areas. The images were compressed to 0.45 bpp before decompression, yielding a 12 dB difference in PSNR.

In another example, the image tested is shown in Fig. 15.20. It is a typical compound
Figure 15.17: Test images: “graphics”, “compound1”, “baby”, “wine”. The contents range from purely pictorial to purely graphics images.
Figure 15.18: PSNR difference plots compared to the PSNR achieved by JPEG compression. The imaginary line at 0 is the JPEG reference performance. The solid line is for MRC compression using simple JPEG+MMR+JPEG and the optimized thresholding segmentation algorithm with $Q_d = 1$; The dashed line is the same but with the fast variance-based thresholding algorithm. The dotted line is for compressing the whole image using single plane JPEG 2000. Image name is noted for each plot set.
Figure 15.19: Reconstructed images after compression at 0.45 bpp. Left: MRC compressed, optimized thresholding segmentation method (PSNR is 37.49 dB). Right: JPEG (PSNR is 25.33 dB).

Our favorite is this picture of us aboard the 'Top Hat', which I have sent into this letter using some really neat advanced digital imaging technology on my home computer. We will ship the next to you on a CD-ROM soon. Wishing you the best.

Love,

Susan
January 31, 2001

Dear Mom and Dad,

How are both of you doing? I thought I would drop a line to say hi. Fanny, little Danny, and I are doing well. As you can see by the picture, little Danny isn’t quite so little! Isn’t this letter really great? I took a picture of Danny that was on a Kodak PhotoCD, and I merged it onto this letter using my computer. I then printed the letter using a color inkjet printer I just bought...

Danny’s wearing the gorgeous **BLUE** sweater you gave him last time you were visiting. It just brings out the **RED** in his lips and cheeks. He definitely gets his good looks from his mother!

Take care of yourselves and write soon.

Love,

Michael

---

color image, with graphics, text and pictures. After compression at a 70:1 ratio, using JPEG, JPEG 2000 and MRC, the decompressed images are shown in Fig. 15.21 and in Fig. 15.22. In Fig. 15.21 it is shown pieces of the text region compressed with different methods, while Fig. 15.22 shows the same for a pictorial region. Note that MRC clearly yields better image quality at both regions for the given compression ratio.

### 15.5 MRC within JPEG 2000

JPEG 2000 is a newer compression standard designed to upgrade the original JPEG [21]. It is based on wavelet transforms and contextual coding of bit planes, achieving a near state-of-the-art performance by the time it was devised. The standard contains 12 parts, but only two of them are directly relevant to us here: parts 1 and 6. In JPEG 2000 part 1 it is defined the core decoder, i.e. the basic compression algorithm that everyone needs to support. For example, part 2 contains extensions, while part 3 deals with Motion
Figure 15.21: Enlarged portion of the text region of original and reconstructed images at a compression ratio of 70:1. Top left: original; top right: JPEG baseline; bottom left: JPEG 2000; bottom-right: MRC using JPEG+MMR+JPEG. Portion size is 256x256 pixels.
Figure 15.22: Enlarged portion of a pictorial region of original and reconstructed images at a compression ratio of 70:1. Top left: original; top right: JPEG baseline; bottom left: JPEG 2000; bottom-right: MRC using JPEG+MMR+JPEG. Portion size is 256x256 pixels.
images, but Part 6 is the one directly related to document compression since it defines a compound image file format. In effect, JPEG 2000 part 6, defines an MRC format within JPEG 2000 [22]–[24]! While JPEG 2000 core files (implicitly part 1) are known as JP2, those for part 6 are known as JPM files (the M for multilayer). We will refer to JPM as a short for JPEG 2000 Part 6 and to JP2 as a short for the core coder in Part 1.

JPM files allow for multipage documents, where each page can be made of a number of layout objects. Actually, JPM employs the concept of page collections wherein a number of individual pages are referenced together. A page collection is in effect a list of pointers to individual pages or to other page collections. An example is a book, where the main page collection points to the chapters’ page collections which point to the sections’ page collections which point to the individual pages. The ordering of pages in the document is defined by the topmost page collection and by those collections pointed by it.

Each page is imaged following the soft mask multilayer model in Sec. 15.2.3. It allows for planes with arbitrary size, resolution and position, which are properly scaled and positioned before imaging. It contains a background and \( N \) mask+image pairs known as Layout Objects. Each layout object contains or assumes a mask and an image layer, e.g. if there is no image associated with the mask, a base color is specified and used. Each layer can be compressed using one of a number of coders including JP2, JPEG, JBIG-2, MMR, etc.

JPM inherits several benefits from the JPEG 2000 family such as the file format and the associated metadata support (e.g. XML). However, since JPM can contain a multipage document, metadata can be associated to a page or group of pages or even to the constituent MRC image layers. All references can be self contained within the same JPM file or can be a remote object. Each MRC layer would contain an URL, an offset, and a length. Note that pages or page collections can be referenced remotely.

The description given here of JPEG-2000 Part 6 multilayer support is very incomplete and can be found in much more detail elsewhere.

A successful MRC compression can only be achieved by an efficient segmentation strategy, as we have discussed. Since JPEG 2000 uses wavelet compression, which has no block structure, one shall use a general segmentation. Recently, an efficient segmentation for JP2-based MRC has been proposed [30], which is the follow-up of a block-based method [31]. DjVu [12] has a built-in segmenter which has been tested in products, typically for very high compression scenarios. There is also multilayer products such as Digipaper [25] and LuraDocument [26] which employ general (non-block-based) segmentation. The goals of a segmentation algorithm can be either to select objects as to facilitate the image representation, or if compression is the main goal, then one may remove sharp edges from the FG-BG planes and move them into the masks. A possible strategy is to find background-contrasting, contiguous, uniform-color objects and move these object shapes onto Layout Objects.

**JP2-based JPM**

Even though any of the many coders available can be used to compress each plane, we suggest using solely JP2 for compression. This is possible because JP2 allows for the compression of either gray-level or binary images. In order to compress binary images with JP2 one may set the number of wavelet levels to 0 and the bit-depth to 1. This will essentially skip the wavelet transform and compress the binary data with the context-driven arithmetic coder which is typically used to compress the wavelet bitplane in JP2.
Hence, the mask is lossless compressed. One might see in the literature comments to the effect that the JP2 binary coder has inferior performance than other binary coders such as MMR and JBIG-2. However, the difference is not constant and is very small. Furthermore the mask plane does not spend too many bits in MRC anyways, so that JP2 is a very adequate substitute for any other binary coder in a JPM context.

We can use a 3-plane decomposition: FG, BG, and the binary mask. The FG plane will have the text colors and can be highly compressed with JP2. The BG plane contains the paper background and pictures and is also compressed with JP2, but under moderate compression. The mask plane is compressed without loss with JP2, as described above. Thus, only the JP2 compression engine is needed to compress the JPM file. Same for decompression. As an example, if we use the compound image and mask shown in Fig. 15.14, we compressed the image using JP2-based JPM, obtaining the RD curves shown in Fig. 15.23. We also compared JPM with JP2 single plane compression. Note the large disparity between JP2 and JP2-JPM: a staggering PSNR difference, beyond 10dB! JPM is indeed an efficient compound image compression scheme.

15.6 Conclusions

In this chapter we tried to provide an overview of predominant techniques for the compression of raster compound documents. We emphasize the term “raster” since we did not discuss any techniques for images in vector form. For raster images, the most important technique of all is the MRC model, mostly because it became an international standard. There are a number of variations to the MRC model as well as unrelated techniques, many of them attaining a very good performance. Nevertheless the MRC model is the predominant technique not only because it is already a standard, but also because it achieves very good RD performance.

The main drawback of MRC (as well as any mixed mode compressor) is the need for reliable segmentation. We have shown how to overcome this problem and implement efficient segmentation for a simple JPEG-MMR-JPEG scheme. This method is block-based and would not work unless we use block based compression for the FG-BG planes.
There is ongoing work for MRC-driven document segmentation for the general case. For the general case, the JPM profile for JPEG 2000 provides an excellent vehicle for MRC compression (even allowing the JPEG-MMR-JPEG case). In JPM, documents can be efficiently compressed using only a JP2 engine for all planes.

Compound documents stress compression techniques. However, the tools we have at our disposal make it quite manageable to compress complex documents to very low bit rates with excellent quality. That enables a number of applications ranging from archival to web-based document retrieval.

We caution the reader for the fact that this chapter is merely introductory. We encourage the reader to seek the references for more detailed discussions and descriptions on MRC and on compound document compression.

References


