Fast segmentation of the JPEG compressed documents

Ricardo L. de Queiroz
Reiner Eschbach
Xerox Corporation
800 Phillips Road M/S 128-27E
Webster, New York 14580
E-mail: queiroz@wrc.xerox.com

Abstract. *We present a novel technique for segmentation of a JPEG-compressed document based on block activity. The activity is measured as the number of bits spent to encode each block. Each number is mapped to a pixel brightness value in an auxiliary image which is then used for segmentation. We introduce the use of such an image and show an example of a simple segmentation algorithm, which was successfully applied to test documents. The document is segmented into characteristics regions labeled as background, halftones, text, graphics, and continuous tone images. The key feature of the proposed framework is that the desired region can be identified and cropped (or removed) from the compressed data without decompressing the image.* © 1998 SPIE and IS&T.
[S1017-9909(98)01002-2]

1 Introduction

Scanned documents often demand large storage space in memory, since the resolution of scanners, monitors, and printers steeply increased in the past few years. In order to save storage space, compression became an important part of scanning and printing systems. A typical scan of an 8.5 $\times$11 in. page at 600 pixels per inch (ppi) generates around 33 million pixels (each pixel comprising multiple bytes, depending on the color space and bit-depth used). As it is common to process several documents at a time, large cost savings arise from compressing the documents.

Typical lossless (or reversible) coders often attain a compression ratio of 2:1 or less for most natural (noisy) images. Thus, lossy algorithms are employed, trying to balance the compression ratio and the image quality. The joint photographic expert group (JPEG) standard coder is widely used for lossy compression of still images. A comprehensive discussion of the standard, along with the recommendation text itself, can be found in Ref. 1.

JPEG has several modes of operation for different applications. For simplicity, we discuss the most popular mode, known as baseline JPEG,[1] and concentrate on the luminance component of color images. For the purpose of this article, we ignore header formats, restart markers, and byte stuffing procedures,[1] since they are not relevant for the discussion. With these restrictions, the presentation is greatly simplified without compromising the spirit of the proposed algorithm. We employ the term JPEG to designate baseline JPEG, unless otherwise stated.

In most cases the image is only available in compressed format. If one wants to process the image, it is often decompressed, and space-domain processing is applied to the reconstructed image.

In any segmentation algorithm two steps must be followed. First, the image is analyzed and suitable data is extracted from it. Second, a segmentation algorithm is applied to the extracted data. As an optional last step, one may apply the segmentation map data to the image in order to facilitate processing or to crop out the desired document regions. Applications include: selective post-processing, background suppression, recompression, automatic feature analysis, and extraction, etc.

1.1 *Objective*

The objective of this article is to present techniques that allow the segmentation of JPEG-compressed images without decompressing them (see Fig. 1). The resulting algorithm is simple and fast in order to save memory and computation. We address the segmentation of a document into specific regions such as those containing halftones, text, continuous-tone pictures, graphics, etc., without decompressing or buffering the whole image. Once the necessary information is quickly extracted from the compressed image, a simple segmentation algorithm is applied to select the desired regions and to generate a mask indicating the blocks pertaining to the selected region. Given this mask, a specialized system operates on the compressed data, extracting subimages or replacing regions, without decompressing the data.

The segmentation algorithm is a simple example intended to support the framework of segmenting in compressed domain and can be replaced by a more sophisticated approach. Also, it is not object oriented, as only pixelwise operations such as linear or simple morphological filters are applied.

Paper 97-070 received Jan. 6, 1997; revised manuscript received Mar. 24, 1997; accepted for publication Sep. 22, 1977.
1017-9909/98/$10.00 © 1998 SPIE and IS&T.

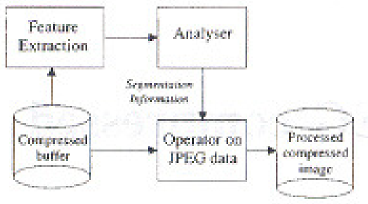


Fig. 1 Objective of the article: a system that can easily extract information from a JPEG image, analyze the information, and process the compressed stream, without decompressing the image.

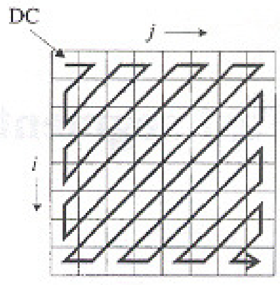


Fig. 3 Zigzag scanning order for an 8×8 block.

1.2 *Organization*

Section 2 presents an overview of JPEG and discusses options for extracting data from the compressed stream. Section 3 presents the proposed technique which is based on the extraction of encoding cost maps (ECM) from the compressed data as a facilitator for the segmentation process. Section 4 presents an example of a simple and efficient segmentation algorithm to be applied to the ECM, which is then used to segment the compressed data. Examples are shown and the method is discussed in detail. Section 5 is concerned with the operation of extracting selected regions from the compressed stream, or with the operation of *deleting* selected (or unselected) regions. Finally, concluding remarks are presented in Sec. 6.

2 Background

2.1 *JPEG Overview*

JPEG is implemented through a sequence of operations as shown in Fig. 2. The image is divided into blocks of 8 ×8 pixels and the blocks are grouped into minimum coding units (MCUs). If the image dimensions are not integer multiples of the dimensions of a MCU, the image may be padded until fitting the desired size. The real image size is conveyed in the header so that the decoder can appropriately crop the resulting image. Color images are frequently converted to a luminance/chrominance color space[1] such as YCrCb, YUV, CIELab, etc., and the chrominance signals are commonly downsampled by a factor of 2 in each direction. In our case, we will concentrate on the luminance component for the sake of simplifying the presentation. For a monochrome image, the MCU consists of only one block.[1] Chrominance information could be incorporated to obtain a more precise algorithm, however, this matter is outside the scope of this article.

The block is transformed using the discrete cosine transform (DCT).[1,2] In JPEG, the DCT is defined by the 8×8 matrix **D** whose entries are

$$d_{ij}^{M}=\frac{1}{2}k_i\cos\left(\frac{(2j+1)i\pi}{16}\right), \tag{1}$$

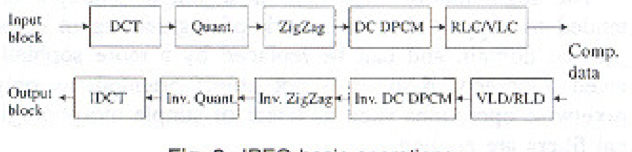


Fig. 2 JPEG basic operations.

where $k_0=1$ and $k_i=1/\sqrt{2}$, for $1\leqslant i\leqslant 7$. Let the block of 8×8 pixels be denoted by **X**, while the transformed block is denoted by **Y**. The separable transform and its inverse are performed as

$$\mathbf{Y}=\mathbf{D}\mathbf{X}\mathbf{D}^T,\quad \mathbf{X}=\mathbf{D}^T\mathbf{Y}\mathbf{D}. \tag{2}$$

The DCT matrices are highly structured and fast implementation algorithms exist.[2]

The subsequent quantization step involves simple unbounded uniform quantizers. A uniform quantizer with step size q_{ij} is applied to each of the 64 transformed coefficients (y_{ij}), generating quantized numbers c_{ij}. Quantization and inverse quantization in JPEG are defined as

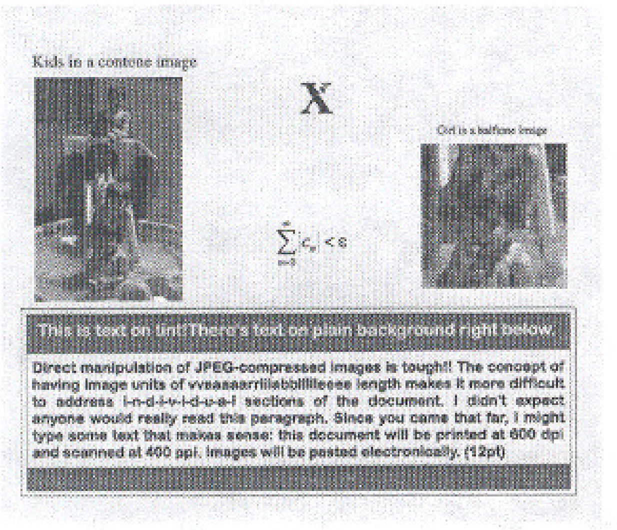
$$c_{ij}=\mathrm{round}(y_{ij}/q_{ij}),\quad \hat{y}_{ij}=c_{ij}q_{ij}. \tag{3}$$

The step sizes are stored in a table which is transmitted along with the compressed data. In case multiple images are transmitted, there is a provision to transmit all the tables upfront.[1] Example (default) tables for luminance and chrominance are given in the JPEG draft and the example quantizer steps for luminance are

$$\mathbf{Q}_l=\begin{bmatrix}16 & 11 & 10 & 16 & 24 & 40 & 51 & 61\\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55\\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56\\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62\\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77\\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92\\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101\\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99\end{bmatrix}. \tag{4}$$

The two-dimensional (2D) array of quantized coefficients c_{ij} is converted into a 1D vector through zigzag scanning, which organizes the data following the path shown in Fig. 3. The DC component (c_{00}) is the first element of the vector and is replaced by the difference between itself and the DC component of the previous block. This resulting vector is losslessly encoded using a combination of run-length coding (RLC) and variable-length coding (VLC). The details of the combination of RLC/VLC can be found in Ref. 1. The decoder reconstructs the vector, accumulates the DC value, reorganize the vector into a 2D block, and performs an inverse DCT to reconstruct the image block.

Throughout the article we use test documents which were compressed using the table in (4) multiplied by a fac-


Kids in a contone image

X

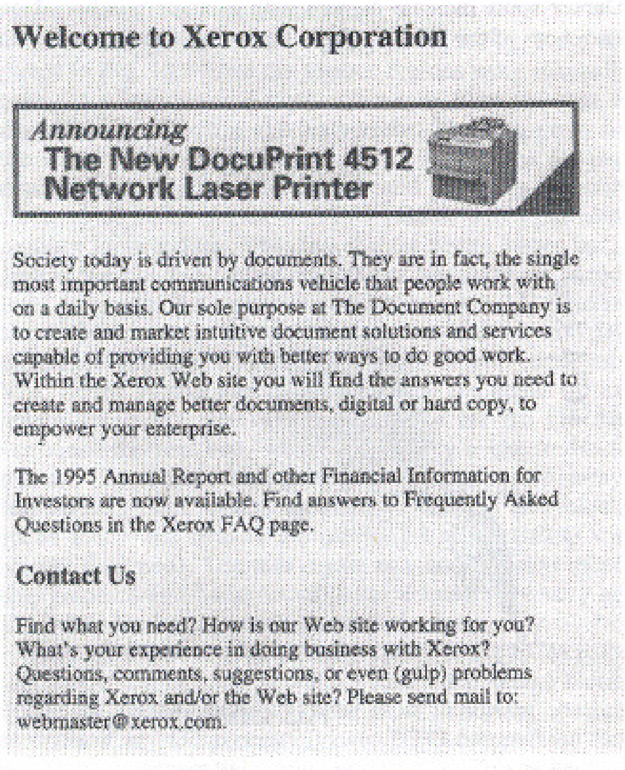
Girl is a halftone image

This is text on tint!There's text on plain background right below.

Direct manipulation of JPEG-compressed images is tough!! The concept of having image units of vvaassarrliiabbliilleees length makes it more difficult to address i-n-d-i-v-i-d-u-a-l sections of the document. I didn't expect anyone would really read this paragraph. Since you came that far, I might type some text that makes sense: this document will be printed at 600 dpi and scanned at 400 ppi. Images will be pasted electronically. (12pt)

Fig. 4 Test image I after decompression at a ratio of 10.7:1.

tor of 1.25, in order to obtain a compression ratio around 10:1. Test image I after decompression is shown in Fig. 4. It contains regions with halftones, text, and continuous tone (contone) images. The document image was originally obtained by scanning documents at 400 ppi and pasting images electronically, before compression. The resolution is 2424×2824 pixels (almost 6×7 in.). Test image II is shown in Fig. 5. Its resolution is 2312×1904 pixels.


Welcome to Xerox Corporation

Announcing
The New DocuPrint 4512
Network Laser Printer

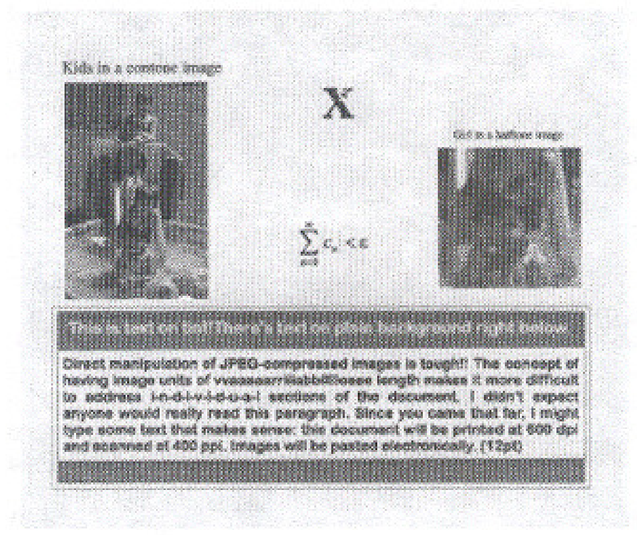
Society today is driven by documents. They are in fact, the single most important communications vehicle that people work with on a daily basis. Our sole purpose at The Document Company is to create and market intuitive document solutions and services capable of providing you with better ways to do good work. Within the Xerox Web site you will find the answers you need to create and manage better documents, digital or hard copy, to empower your enterprise.

The 1995 Annual Report and other Financial Information for Investors are now available. Find answers to Frequently Asked Questions in the Xerox FAQ page.

Contact Us

Find what you need? How is our Web site working for you? What's your experience in doing business with Xerox? Questions, comments, suggestions, or even (gulp) problems regarding Xerox and/or the Web site? Please send mail to: webmaster@xerox.com.

Fig. 5 Test image II after decompression at a ratio of 10.2:1.


Kids in a contone image

X

Girl is a halftone image

This is text on tint!There's text on plain background right below.

Direct manipulation of JPEG-compressed images is tough!! The concept of having image units of vvaassarrliiabbliilleees length makes it more difficult to address i-n-d-i-v-i-d-u-a-l sections of the document. I didn't expect anyone would really read this paragraph. Since you came that far, I might type some text that makes sense: this document will be printed at 600 dpi and scanned at 400 ppi. Images will be pasted electronically. (12pt)

Fig. 6 DC map for test image I.

2.2 *Representative Functions*

In the discussion, we assume that the document is scanned at a reasonably high resolution (for good image quality) generating a large amount of data (that is why compression was required in the first place). If we perform segmentation in the space domain after decompressing the image, not only would it represent a large computational effort to decompress the image, but it would also demand a large memory buffer to store the image for segmentation. An alternative is to extract information from the compressed data that enables the segmentation process without using the full image resolution. This information is a function of the data. We analyze some representative functions which are helpful to enable the segmentation.

Apart from decompressing the image, one can extract the DC component of each block and compose an auxilliary image where each pixel represents the DC component of each original 64-pixel block. This auxilliary image is called the DC map of the image. The DC map corresponding to test image I is shown in Fig. 6. One option is to feed the DC map to a segmentation algorithm using it as an *original* image. This will result in a small image (64 times smaller) to be processed, but resolution is compromised. This is because the DC coefficient does not convey information about the intra-block spatial frequency (activity). However, it is easy to obtain the DC map from the compressed bitstream, making it a low-cost alternative.

Another alternative for an auxilliary image is to obtain the DCT coefficients of each block and to compute a measure of AC energy of the block, where the AC energy is defined as

$$E_{\mathrm{ac}} = \sum_{ij,ij \neq 00} c_{ij}^2 . \tag{5}$$

Sometimes the AC energy is taken as the sum of the absolute value of the coefficients, instead, in order to save computation. If, for a block at coordinates (m,n), we compute the AC energy as

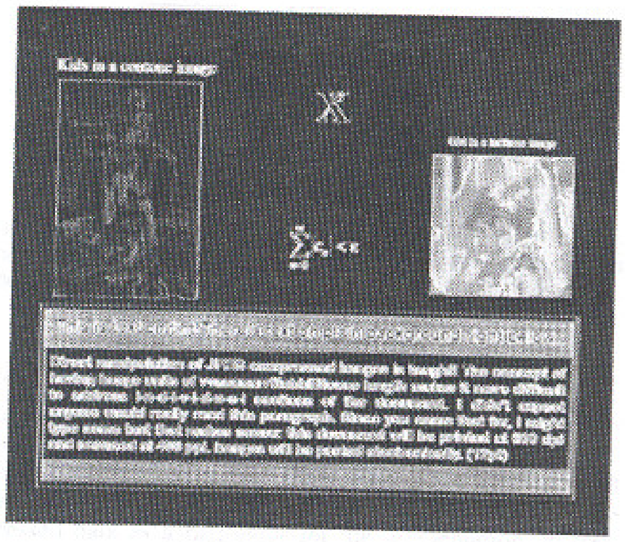


Fig. 7 AC energy map for test image I.

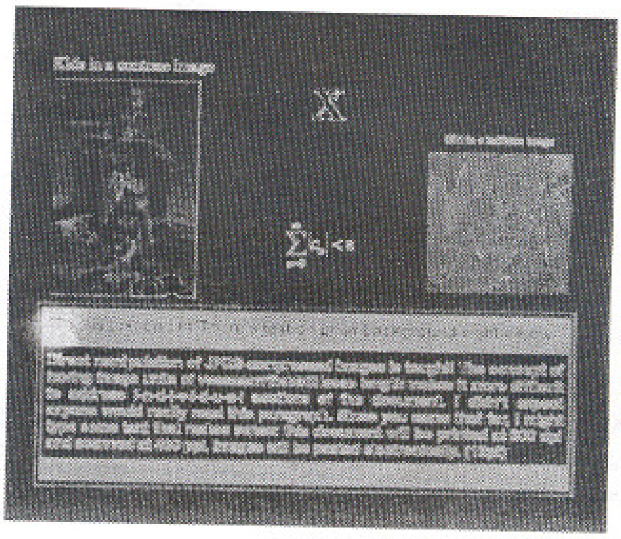


Fig. 8 ECM for test image I.

$$e_{mn}=\frac{1}{63}\sum_{ij\neq 00}|c_{ij}|,$$

the AC energy map corresponding to Fig. 4 is shown in Fig. 7, where entries larger than 255 were clipped to this value.

We look for functions as simple to compute as the DC map, while retaining the information of the AC energy map. Such a function is discussed next.

3 Cost Map

We propose using another measure of activity for a block as an input for the segmentation process. We define the ECM as the number of bits spent to encode each block. The term *cost* derives from the analogy between the cost-benefit and the rate-distortion pairs. The cost of sending a block is the amount of bits spent to encode it. The benefit of sending a block is the decrease in reconstruction distortion yielded by providing the block data to the receiver. In this case, each block of the compressed image maps to an integer number and we can construct a 2D map (image) where each entry (pixel) corresponds to the number of bits used to encode an 8×8 block of the original image. For block coordinates (i,j) we denote the ECM entry as b_{ij}. The ECM corresponding to test image I is shown in Fig. 8. In this image, each pixel corresponds to one b_{ij} entry. The basic reason for using the ECM is because it captures the information regarding intra-block activity, just like the AC energy, but at a much lower computational cost. The following discussion is intended to support this statement.

The ECM computation is virtually the simplest operation one can perform on the compressed stream. Not even the variable length codes have to be fully decoded. Only the code word sizes are important and the decoder does not need to be concerned with reconstructing coefficient sign, offset, etc.

JPEG provides compression for images which fit a model of smooth image regions. Smooth blocks tend to generate low-frequency DCT coefficients only, while active blocks tend to generate high-frequency coefficients too. The smooth blocks are likely to be encoded with few bits while the busy blocks spend more bits to do so. This is the image compression principle behind JPEG which is reflected in the ECM. Edges and texture might produce a large quantity of encoding bits because they produce several nonzero high-frequency coefficients. Since JPEG compression relies on RLC/VLC, a larger number of nonzero coefficients may produce a larger number of encoding bits. A correlation between the AC energy entries e_{ij} and the respective ECM entries b_{ij} is shown in Fig. 9, i.e., the correlation between the images shown in Figs. 8 and 7. Darker spots indicate regions with a larger number of occurrences of the pair (e_{ij},b_{ij}). From Fig. 9, we can see that

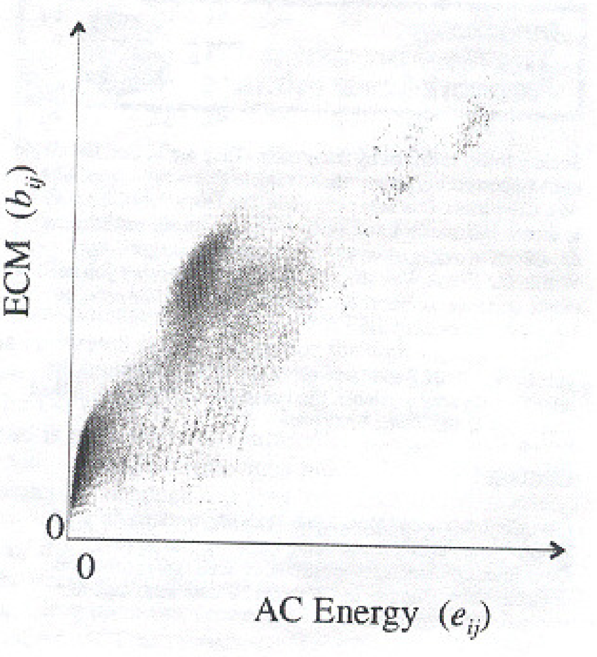


Fig. 9 Correlation between ECM and AC energy map entries, for test image I.

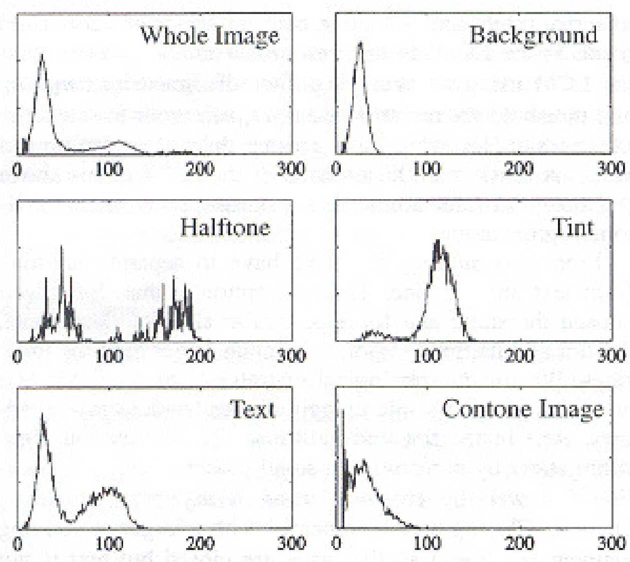


Fig. 10 ECM histograms for test image I.

there is a strong correlation between the two functions. The ECM tends to limit the range of the AC energy entries. This is a nice property which avoids the large range of values found in the AC energy map. Another difference between the two maps lies in the fact that AC energy does not consider frequency information, i.e., coefficients with the same amplitude at different frequencies contribute equally to the energy computation. The same is not true for the ECM because of the zigzag scanning and of the VLC/RLC compression used in JPEG. The samples to be encoded using run-length follow an ordering procedure from low to high frequencies, so that before reaching a high-frequency coefficient, several *runs* are encoded.

ECM histograms for the image shown in Fig. 8 are depicted in Fig. 10, where the various regions were manually segmented by cropping small subimages from the part of interest and computing the histograms. The histogram for the whole image is shown along with histograms for particular image regions. The background region is basically made up of ECM entries with low values, while halftones produce large values. Note that for the rough paper background we can use almost 50 bits per block for its encoding. A purely flat artificial background would generate about six encoding bits per block[1] (two bits for null DC difference and four bits for EOB) using default luminance VLC tables. Active blocks are present in the edges of text letters, equations, etc.[3] The same is true for the tint that surrounds the text box, which is basically a checkerboard pattern. The contone histogram follows an expected decaying pattern: more blocks spend few bits and less blocks spend many bits (this is the pattern for which JPEG was mainly designed). The text region is a combination of active graphics and background: letter contours are very active while the background is a smooth region.

Another important functional property of the ECM is the fact that it can be used to address individual image blocks inside the compressed data.[4] The most important obstacle to perform any fast processing over JPEG-compressed data resides in its sequential compression mode. This is because the number of bits spent to encode each block is variable. In general, one cannot decode a block before decoding the preceding bit-stream. Thus, ECM facilitates cropping operations in the compressed domain.[4] We will discuss this topic later on.

In terms of robustness, the selected function has to enable a segmentation algorithm that works independently of certain compression parameters such as the compression ratio. To investigate this, consider that the sum of all b_{ij} is the number of bits spent to encode the whole image. A b-bits-per-pixel image with dimensions $N_1 \times N_2$ achieves a compression ratio $\mathrm{CR} = bN_1N_2/\Sigma b_{ij}$. If B is the average ECM entry [i.e., $B = \Sigma b_{ij}/(N_1N_2)$], then $\mathrm{CR} = b/B$. As we change the compression ratio, the ECM changes. In order to have a scalable algorithm we expect that as we decrease the compression ratio, the bits are proportionally distributed to all blocks. In other words:

$$B' = \alpha B'' \rightarrow b'_{ij} \approx \alpha b''_{ij}. \tag{6}$$

The above relation does not always hold, but at least we would like to approximate it for regions of the image. Let a region R of the image be composed by one pattern, e.g., halftone, background, etc. Suppose this region has N_R pixels. The average ECM entry in this region is

$$B_R = \frac{1}{N_R} \sum_{ij \in R} b_{ij}. \tag{7}$$

Figure 11 depicts the behavior of the regional averages B_R as a function of the compression ratio. Different compression ratios were obtained by scaling the table in (4). Figure 11 also shows the same plot for a normalized average B_R/B. By inspecting this figure for such a large range of compression ratios we assume

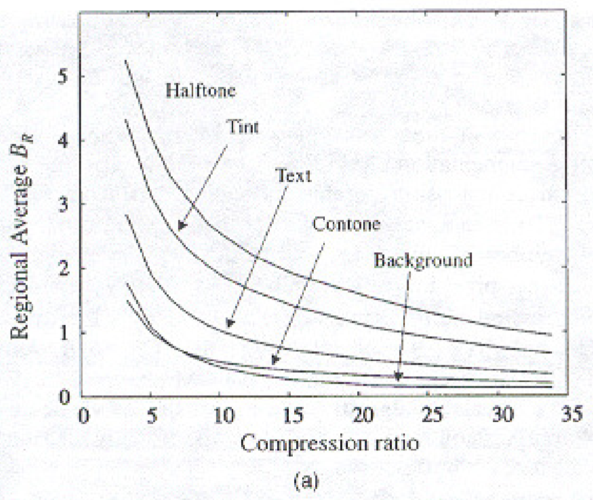
$$\frac{B_R}{B} \approx \text{constant}, \tag{8}$$

which enables the use of an ECM-based segmentation algorithm for a broad spectrum of compression ratios. Roughly, the change in the number of bits spent to encode the whole image is distributed proportionally to the different image regions. Therefore, by setting relative levels in a segmentation algorithm, one can scale these levels according to the compression ratio obtained.

4 Segmentation of the ECM

Various segmentation and classification approaches have been published and most of them are based on examining the decompressed image and perhaps extracting high-frequency information (edges) to localize letters, graphics, etc. See Refs. 5–10 for examples of segmentation. We use the ECM for segmentation because it is simple to generate the data and because of the properties discussed in the previous section. Furthermore, if the ECM is sent as side information no computation is necessary.

As is clear from Figs. 9 and 10, the amplitude of the ECM entries for the different image regions overlap. Different blocks may generate the same ECM entry and the pure histogram analysis of the ECM may not suffice for any robust segmentation algorithm. This problem is common to



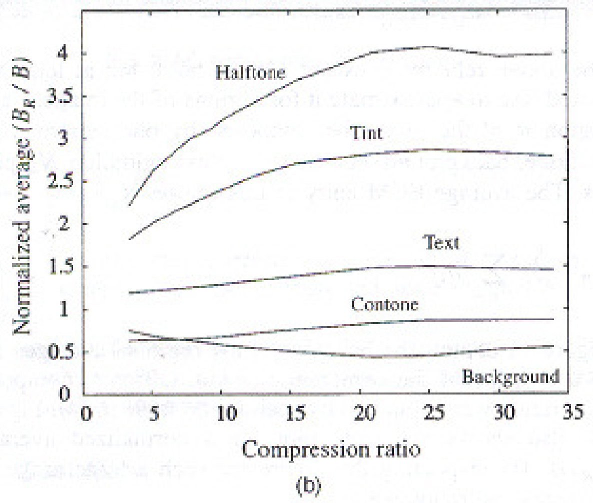


Fig. 11 (a) Behavior of regional average of ECM values B_R as a function of the compression ratio. (b) Same as (a) where B_R was normalized by the overall average B.

all functions discussed in Sec. 2.2. In the example of Fig. 4, we would like to identify the following regions: halftone or tint, contone, text or graphics, and background. Each one has its own characteristics in terms of amplitude and spatial distribution of ECM pixels and we will discuss them in detail.

4.1 *Outline of an Example of Sequential Segmentation*

Next, the basic algorithm is described. We follow a sequential approach in which we first find a particular region, suppress it from the input image, and proceed to find the next region. For each region we analyze its predominant pattern and decide how to isolate the region, using only simple pixel-oriented and morphological operations.

4.1.1 *Halftone*

Regions with very dense concentration of bright pixels in the ECM may indicate a halftone. Text regions produce a more sparse distribution of bright ECM pixels because there is background between the letters. The same is true for graphics, equations, etc. Hence, in order to detect halftone one might look for large concentrations of very bright pixels in the ECM. In this case, for example, we can filter the ECM using an averaging filter of dimensions $n_0 \times n_0$ and threshold the result (threshold t_0) in order to select the filtered samples which are greater than a predetermined value. A mask with the position of the ECM pixels above the threshold value are used as a starting point for the halftone segmentation.

From this starting point we have to separate halftone from text and contone. The assumption is that, for a well chosen threshold and for a particular class of documents, the tint and halftone regions are much larger than the letter sizes. We use morphological operators[11] to eliminate text and contone spots while plugging holes (such as text, shadows, etc.) in the tint and halftone. The elimination algorithm starts by performing a small closing operation (dilation followed by erosion), using a $m_0 \times m_0$ structuring element. Closing may be repeated with a larger structuring element $(m_1 \times m_1)$ so that gaps are closed but text is not fused with other objects. After that, the text and graphics may be fused into small solid objects. If we estimate the letter sizes (in blocks), we can use a morphological opening operator using an element whose size $(m_2 \times m_2)$ is comparable to the text letter size. This will erode the text regions while only reducing the halftone region. The resulting mask undergoes further dilation with a small structuring element $(m_3 \times m_3)$ whose purpose is twofold: (i) it provides a safety margin for the masks and (ii) it rounds up the object boundaries, making them more uniform.

4.1.2 *Background*

The background (paper) is a region where the predominant pattern is composed of dark ECM pixels, among few lighter ones. The average appearance of the region is dark and one may use the same method used in the first step of the procedure used to find halftones. If we filter the ECM using an $n_1 \times n_1$ averaging filter and threshold the result, filtered pixels below that level (t_1) are considered background. However, this mask may be polluted with several smooth contone blocks. In this case, it may be advantageous to also use the DC map of the image. This problem is also shared by segmentation algorithms that use the AC energy map. If we select the blocks where the filtered ECM is below the threshold t_1 and the filtered DC (using an $n_1 \times n_1$ averaging filter) is above threshold t_2, we may obtain a mask containing a much better description of the background.

4.1.3 *Contone*

If we suppress the background and halftone regions from the ECM, we are left with regions containing contone and text (graphics). Let the mask M be composed of the complementary set of the union of the halftone and background set, i.e., M is made of whatever is left after deleting halftone and background. We are left with the task of separating M into contone and text regions. Again, contone regions are assumed to be much larger than the text letters. We now proceed with an operation which is similar to the one used to find halftones. However, now, we do not have a filtered and thresholded mask to work with. Therefore, it is not necessary to close gaps in the contone region. Furthermore, we have to be more aggressive with the text size estimation because the background estimation itself was conservative. As a result we can use directly a morphological opening using a larger structuring element of size $m_4 \times m_4$ which should be larger than the text letters and would ensure that text and graphics are completely eliminated. At the end, we perform a dilation for the same reasons as in the halftone case (safety margin and uniform boundaries) using an $m_5 \times m_5$ structuring element.

4.1.4 *Text and graphics*

If we suppress the contone mask from mask M, whatever is left is considered to be text and graphics. In a variation of the algorithm one may re-address the remaining information to be sure small patches of what otherwise would be considered halftone or contone data were not left there simply because they were eroded along with the text.

4.2 *Setting Parameters*

The segmentation algorithm was formulated as a function of several parameters which must be adjusted according to: document class, typical font sizes, resolution, paper, scanner, gamma correction, compression ratios, etc. However, in light of the following discussion it may be easier to set the parameters for a particular class.

We have to set up ten parameters: thresholds t_0, t_1, and t_2, filter sizes n_0 and n_1, and the structuring element sizes m_0 through m_5.

We want to select the smallest filter sizes in order to save computation and to avoid excessive blurring of the ECM pixels. Then, we can set $n_0 = n_1 = 3$. In order to set up the rest of the parameters we have to calculate the following:

(i) average bit-rate (B) in bpp,

(ii) maximum letter size S_L of typical text (in blocks),

(iii) regional average bit-rate (B_R/B),

(iv) average paper brightness intensity in background regions (I_B),

(v) minimum background brightness intensity value $I_{\min}$, so that 95% of the background image pixels are above $I_{\min}$.

The values of I_B and $I_{\min}$ are a direct function of paper type and scanner settings. If we want to separate tint and halftones from text, we set a threshold lying between the normalized regional averages of the halftone and text regions. This threshold is multiplied by B to denormalize the number and the result is multiplied by 64 since there are 64 pixels in a block. Hence,

$$t_0 = 64B\frac{1}{2}\left(\frac{B_{\text{halftone}}}{B} + \frac{B_{\text{text}}}{B}\right), \tag{9}$$

where it should be noted that we use the lowest frequency halftone. In the example in Fig. 11, we might use the tint region. The same concept can be used to set t_1, where we want to detect background. We can set a threshold between the normalized regional averages of the background and text regions. Thus,

$$t_1 = 64B\frac{1}{2}\left(\frac{B_{\text{background}}}{B} + \frac{B_{\text{text}}}{B}\right). \tag{10}$$

The intensity threshold for background can be directly set as $t_2 = I_{\min}$. The size of the binary structuring elements for morphological operations is largely decided by empirical tests. In order to detect halftones we may chose m_0 small enough not to fuse text and halftones and large enough to close gaps in the halftone areas. A starting point is to use $m_0 = 3$ and $m_1 = 0$ (not use) and change the parameters based on experimentation. m_2 has to be large enough to remove text letters. Since the letters were already reduced (eroded) by filtering, we may choose m_2 as the largest integer number smaller than S_L. The final step in finding halftones is a mask dilation which can use $m_3 = 3$ or $m_3 = 5$, since this step is just a precaution measure to avoid small isolated *holes* inside the selected regions. m_4 follows the same reasoning used to choose m_2, however, since there is no erosion by filtering, a safe value is to select m_4 as the smallest integer larger than S_L. m_5 is chosen in the same way as m_3.

4.3 *Segmentation Example*

We now apply the segmentation algorithm to our test image. The original image has 8 bpp and was compressed at CR=10.68, so that $B = 0.748$ bpp. A 12 pt object at 400 ppi uses the height of 66.4 pixels or $S_L = 8.3$ blocks. For the particular scanner settings and paper used, we found that $I_B = 235$ and $I_{\min} = 220$. By inspecting Fig. 11 we set B_R/B to be 2.3, 1.25, and 0.5 for tint (lower frequency halftone), text and background regions, respectively.

From these parameters, we find $t_0 = 85$, $t_1 = 42$, $t_2 = 220$, $m_2 = 7$ and $m_3 = 9$. The other parameters were chosen as recommended, i.e., $n_0 = n_1 = m_0 = 3$, $m_1 = 0$, and $m_4 = m_5 = 5$. The filtered and thresholded ECM (using t_0) is shown in Fig. 12(a), while Fig. 12(b) shows the mask for halftone regions after morphological processing. If we use only the ECM to find the background regions, by filtering and comparing the ECM to t_1, we obtain the mask shown in Fig. 12(c), while the preferred algorithm (also using the DC map) resulting in the mask shown in Fig. 12(d). If we remove the pixels masked by Figs. 12(b) and 12(d) from the ECM image shown in Fig. 8, the remaining ECM entries of interest are marked by mask M which is shown in Fig. 13(a). From M we obtain the mask for contone regions which is shown in Fig. 13(b). The remaining entries comprise the text and graphics mask which is shown in Fig. 13(c). This mask can be optionally dilated for safety as shown in Fig. 13(d).

As a second example, we try to find a small halftone object in test image II. For this image, $I_B = 215$ and $I_{\min} = 200$. The masks for background, text and graphics, and halftone are shown in Fig. 14.

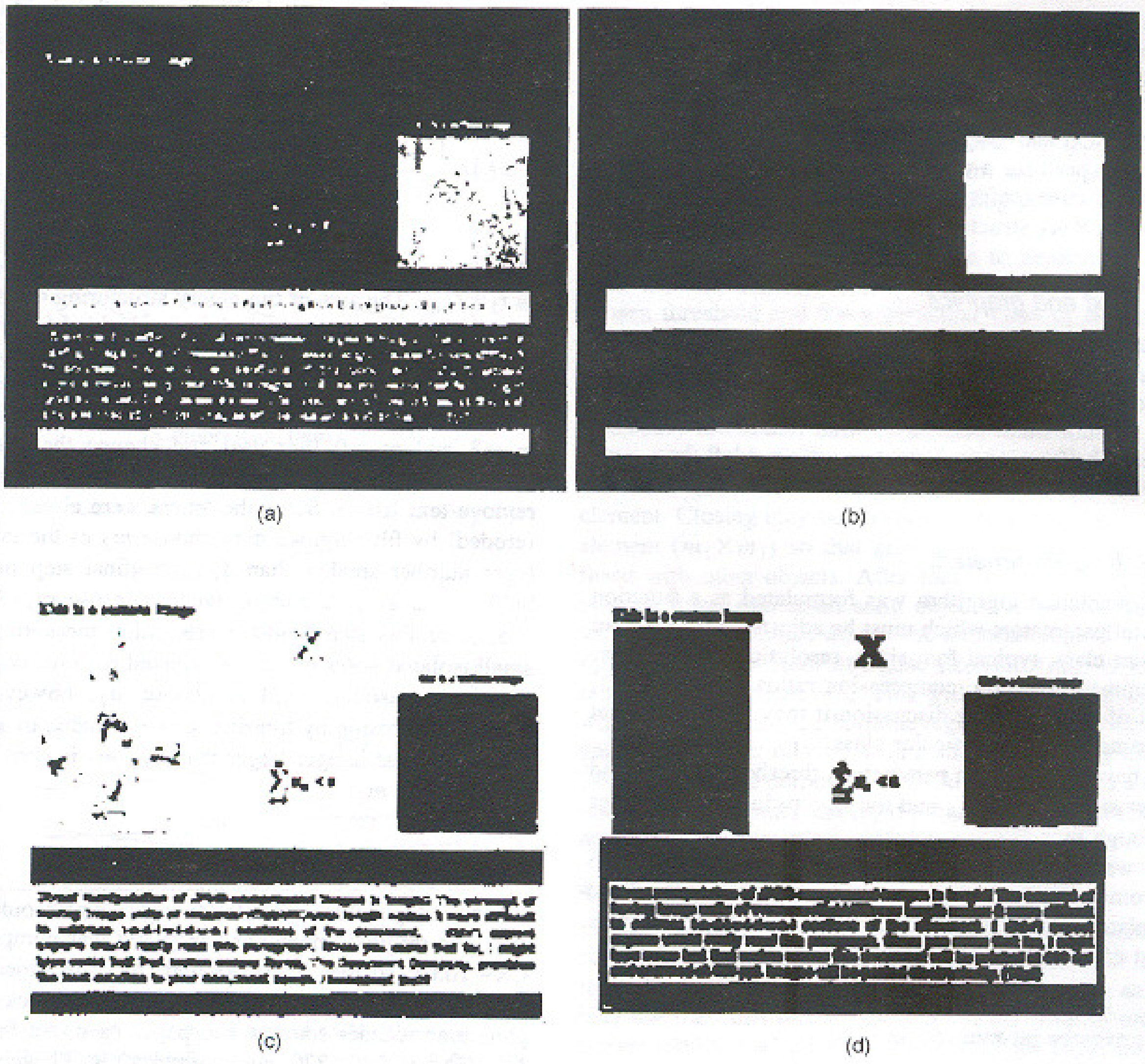


Fig. 12 Segmentation masks for test image I: (a) filtered and thresholded ECM values; (b) halftone; (c) background using only the ECM; and (d) using also the DC map.

4.4 *Algorithm Limitations*

The algorithm as described in this article is intended to demonstrate the feasibility of segmenting in the compressed domain, rather than to offer an optimized segmentation algorithm. Consequently, the current algorithm has to be supervised. The parameters are functions of the input data, e.g., resolution, scanner settings, and document substrate. Just like in other segmentation algorithms, one can derive an automatic algorithm, based on learning and heuristics. This algorithm is intended as an example to support the use of ECM for segmentation of compressed images. Nevertheless, for fixed conditions (given paper, scanner, and target documents) it can be very effective if some calibrations can be done off-line. The current algorithm is strictly pixel-based and therefore does not offer any high-level recognition, such as table versus equation recognition. Extending the presented method to higher level recognition is beyond the scope of this article.

The key to the algorithm is its speed. It takes less time to browse the JPEG document and apply simple operations to the ECM than it takes to decompress the full image. Therefore, regardless of the segmentation algorithm applied to a decompressed image, the proposed method will always be faster than any strategy involving the processing of the decompressed image. Furthermore, the mask enables cropping and pasting in the JPEG domain (as we will discuss next), and a space-domain algorithm would have to require full JPEG recompression.

The algorithm will no longer be robust for very low resolution documents, where details of text letters may be contained inside single blocks. In other words, in this case, the resolution of the ECM (and of the DC map, as a matter of fact) will be too low for reliable determination of the characteristics of the region. For example, in a 300 ppi image, the ECM will have resolution of 37.5 ppi, i.e., about 2 pt per ECM pixel. An 8 pt text would still span several blocks, therefore, several ECM pixels. If the resolution drops to 75 ppi, most text letters of 8 pt will be contained in one block, therefore falling below the minimum discernible resolution of the algorithm.

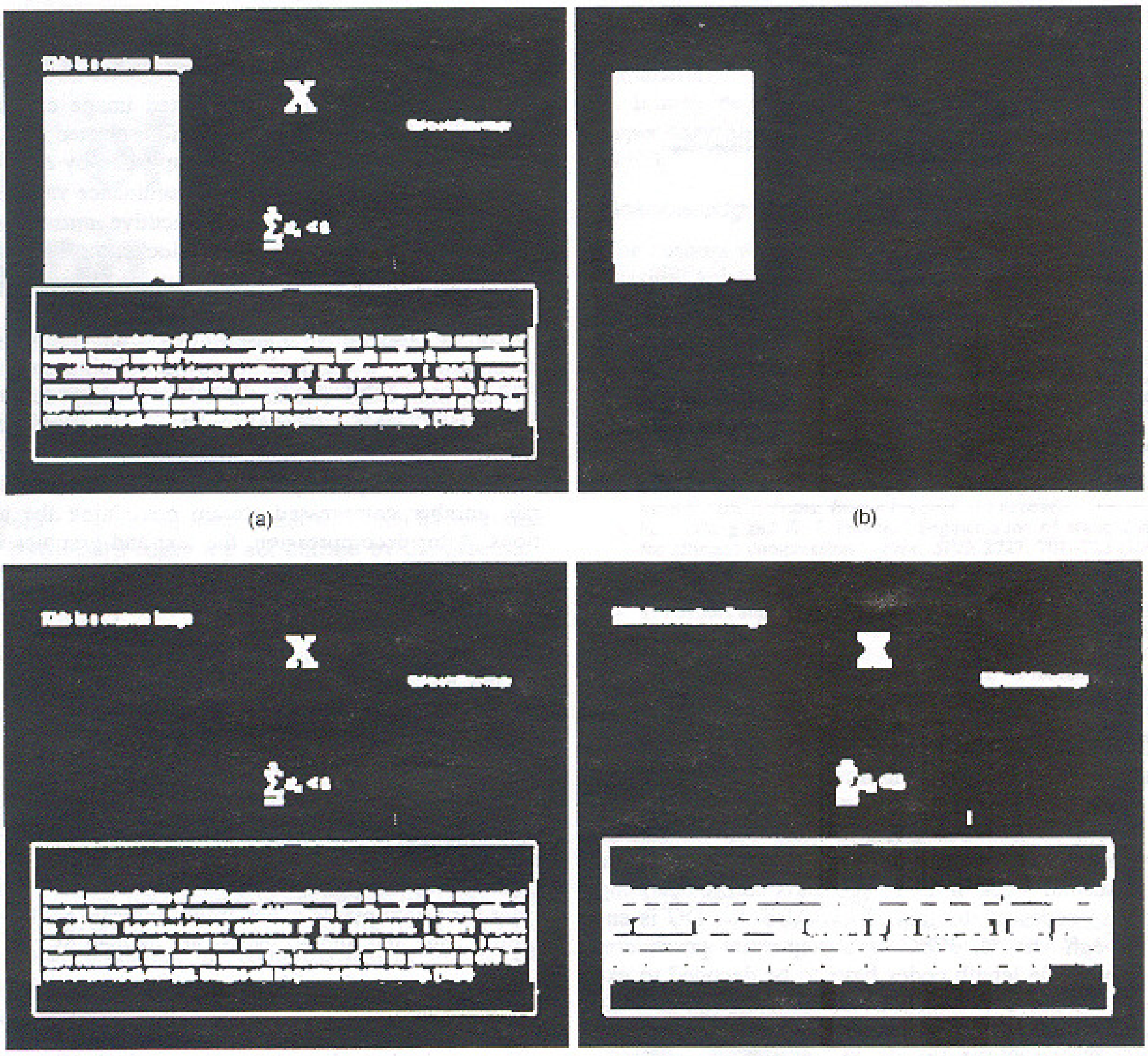

(a) (b) (c) (d)

Fig. 13 Segmentation masks for test image I: (a) remaining ECM pixels after suppressing background and halftone; (b) contone; (c) text; (d) same as (c) after dilation.

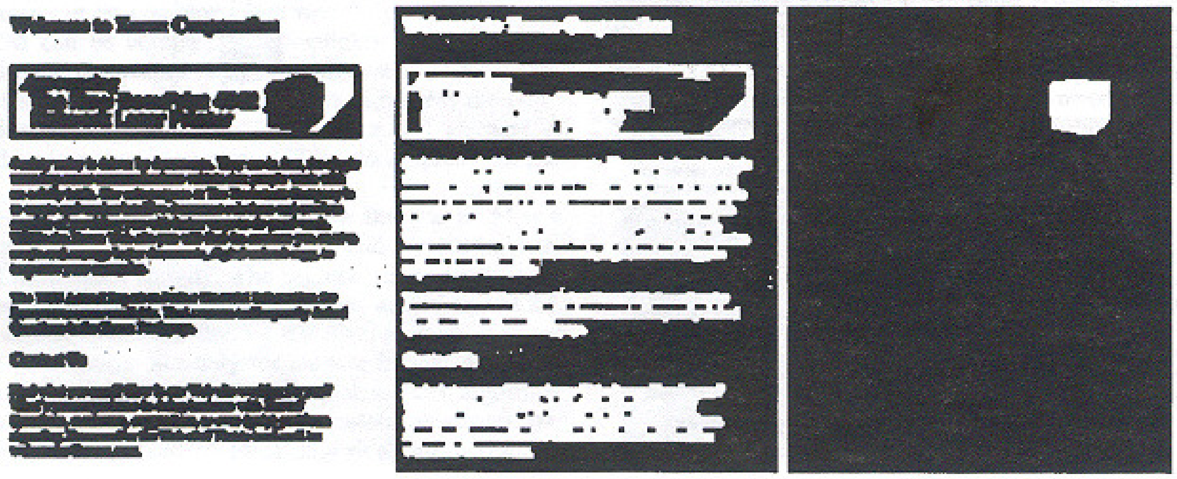
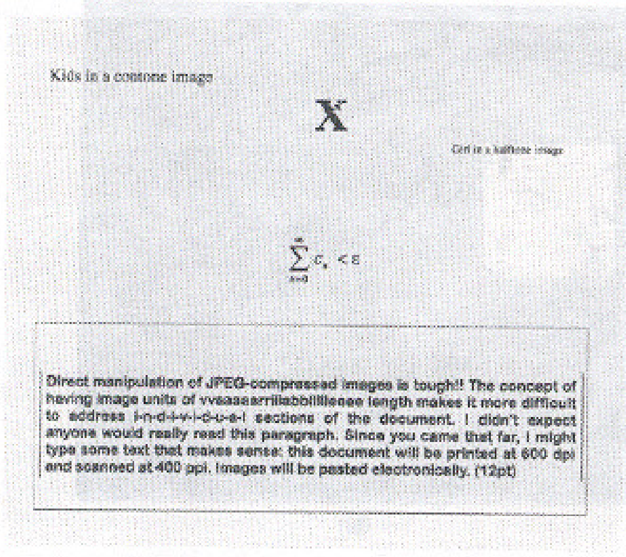


Fig. 14 Segmentation masks for test image II: (left) background; (center) text and graphics; (right) halftone.


Kids in a contone image

X

Girl in a halftone image

$\sum_{i=0}^{n} c_i < \varepsilon$

Direct manipulation of JPEG-compressed images is tough!! The concept of having image units of vvaaaarriiaabbiilliiee length makes it more difficult to address i-n-d-i-v-i-d-u-a-l sections of the document. I didn't expect anyone would really read this paragraph. Since you came that far, I might type some text that makes sense: this document will be printed at 600 dpi and scanned at 400 ppi. Images will be pasted electronically. (12pt)

Fig. 15 Reconstructed image I masked by the undilated text mask. Unused blocks are replaced by gray level $I_B = 235$.

5 On Extracting Selected Regions

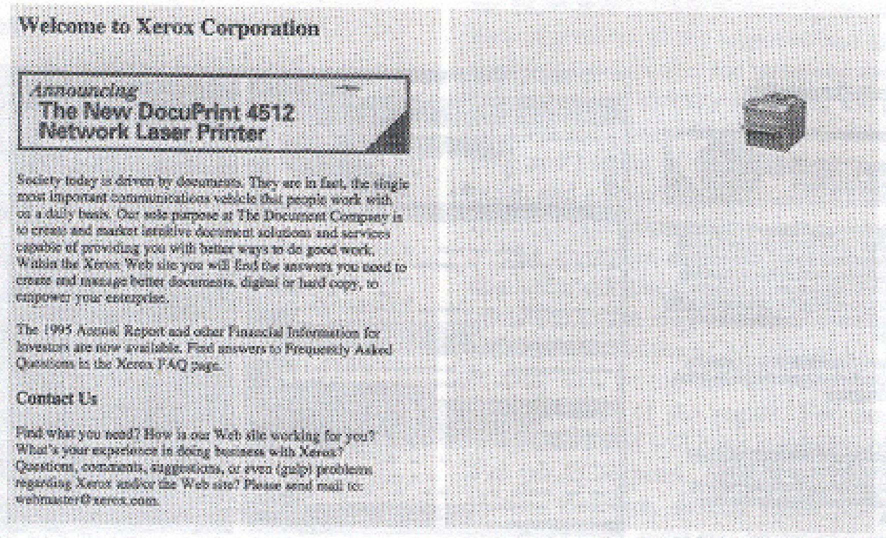
Once the ECM is segmented by binary masks we can extract the necessary information from the compressed bitstream. The ECM and the DC map provide means to access individual blocks in the image. It is clear that because of the compression method one cannot know where the data regarding second block of the image starts before decoding the Huffman codes of the first block. Also, the DC is encoded through DPCM. JPEG is a sequential procedure. However, variable length codes have to be decoded to extract the ECM and DC map. As we know the size of each block and the real DC component, we can quickly *seek* the file to the start of the nth block and grab the exact amount of bits pertaining to the nth block. Thus, no further decoding operations are needed and one can easily crop the desired regions without decompressing the image. In this case, only the selected data need to be extracted as long as the resulting image is rectangular.

On the other hand, the extracted image can have the same size of the original, but with unselected regions suppressed. This can be easily accomplished by replacing the pixels in those regions by a fixed luminance value I_B. This will form several streams of consecutive *unused* blocks. In a stream of consecutive unused blocks, the DC difference of the first block is encoded as the value of $round(8I_B/q_{00})$ minus the quantized value of the previous block. For the remaining consecutive unused blocks, the compressed data is replaced by a standard bit string defining: null DC difference plus EOB (which is 001010 using default luminance VLC tables). For example, if we mask the reconstructed test image I with the text mask, (replacing the unused blocks by the predetermined DC value I_B) we obtain another compressed stream containing the text portions. After decompression, the text-and-graphics image is shown in Fig. 15. As another example, test image II, masked by the maps in Fig. 14, is shown in Fig. 16.

In Fig. 15, one should note the ringing around the text edges. This is not caused by the compression method, but by the background replacement. Since the paper surface was not perfectly smooth it created a seam between the regions with artificial and natural background regions. However, these artifacts can be easily removed by applying a σ-filter only at the region boundaries. The σ-filter behaves mostly like a linear convolution filter, except that it just includes samples whose levels are within $\pm\sigma$ levels from the center pixel. We also want to only activate the filter for bright pixels which might indicate background regions. Thus, the filtered values are defined by

$$\tilde{x} = \text{mean}\{x_{ij} | ij \in W, |x - x_0| \leqslant \sigma, \ x_0 > I_{\min}\}, \tag{11}$$

where x_0 is the value of the center pixel, W is a window neighboring x_0. In a σ-filter, the estimate of the smoothed value $\tilde{x}$ is the local running average of the neighboring


Welcome to Xerox Corporation

Announcing
The New DocuPrint 4512 Network Laser Printer

Society today is driven by documents. They are in fact, the single most important communications vehicle that people work with on a daily basis. Our sole purpose at The Document Company is to create and market intuitive document solutions and services capable of providing you with better ways to do good work. Within the Xerox Web site you will find the answers you need to create and manage better documents, digital or hard copy, to empower your enterprise.

The 1995 Annual Report and other Financial Information for Investors are now available. Find answers to Frequently Asked Questions in the Xerox FAQ page.

Contact Us

Find what you need? How is our Web site working for you? What's your experience in doing business with Xerox? Questions, comments, suggestions, or even (gulp) problems regarding Xerox and/or the Web site? Please send mail to: webmaster@xerox.com

Fig. 16 Reconstructed image II masked by the respective masks: text and graphics (left) halftone (right). Unused blocks are replaced by gray level $I_B = 215$.

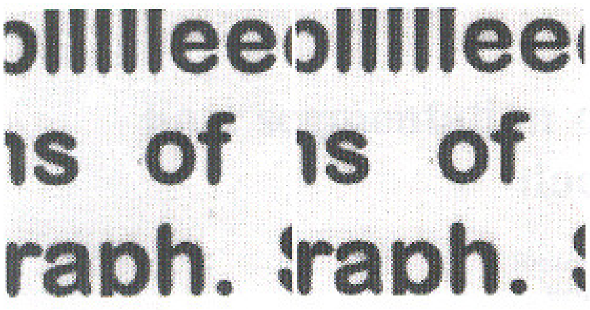


Fig. 17 Stitching problems at background boundaries. The decompressed image is processed by a σ-filter. The filter is programmed to process pixels above a luminance threshold which are located close to the background region. Left: 200×200-pixels part of the reconstructed image; right: same image after processing.

pixels, within a specified local window, which have similar values. Thus, sharp edges are not blurred by the process.

Figure 17 shows a portion of Fig. 15 before and after processing. The same procedure can be applied to all combinations where the background is replaced.

6 Conclusions

We presented a fast algorithm for the segmentation of an image in the compressed domain, in which we extract and process the ECM of a JPEG-compressed document. We apply morphological operations to the ECM in order to find the masks for the regions of interest. These operations are done sequentially. As one region is identified, it is eliminated, simplifying the remaining segmentation tasks. The final binary masks in conjunction with the ECM and DC map are used to address the compressed data and to crop the desired (segmented) regions without decompressing the bit stream. Finally, post processing is discussed to reduce uneven boundaries between original and artificial background in segmented images, after they are decompressed.

The proposed algorithm is very fast and avoids expensive operations such as the forward and inverse DCT. Morphological filtering is not necessarily an expensive process[12,13] and is only applied to a largely reduced image (the ECM). For the simple binary case using square structuring elements they can have very fast algorithms and several stages can be combined into one faster operation. As the ECM can be computed very quickly from the compressed data, the overall segmentation system is very fast. In our tests we were able to segment and process the compressed stream (yielding another compressed stream) in about the same time necessary to fully decompress the image.

The goal in this article was to show that the ECM is a useful tool that allows parsing the data and provides a measure of intra-block activity. The segmentation algorithm is an example to serve as support for the use of the ECM. However, the masks for finding the diverse image regions were quite accurate, not only for the two images presented but for other images tested. Nonetheless, the algorithm needs supervision and is largely dependent on several parameters that have to be set using empirical results. Further research is necessary to improve the robustness of the segmentation algorithm and to automatically set the necessary parameters.

It may be appreciated that the principles used in this paper may also apply to compression schemes other than JPEG.

Acknowledgment

The authors wish to thank Patrick Fleckenstein for the discussions related to this article.

References

1. W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Compression Standard*, Van Nostrand Reinhold, New York (1993).
2. K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Academic, San Diego, CA (1990).
3. R. Eschbach, "Efficient iterative decompression of standard ADCT compressed images," U.S. Patent No. 5,521,718.
4. R. de Queiroz, "Processing JPEG-compressed images and documents," *IEEE Trans. Image Process.* (to appear).
5. H. T. Fung and K. J. Parker, "Segmentation of scanned documents for efficient compression," *Proc. SPIE* **2727**, 701–712 (1996).
6. S. N. Srihari, "Document image understanding," *Proc. IEEE Int. Symp. on Circuits and Systems*, pp. 87–96 (1986).
7. T. Pavlidis and J. Zhou, "Page segmentation and classification," *CVGIP: Graphical Models and Image Process.* **54**, 484–496 (1992).
8. D. Dunn, T. Weldon, and W. Higgins, "Extracting halftones from printed documents using texture analysis," *Proceedings of the International Conference on Image Processing*, Lausanne, CH, 1996, Vol. II, pp. 225–228.
9. K. Murata, "Image data compression and expansion apparatus, and image area discrimination apparatus," U.S. Patent No. 5,535,013.
10. Z. Fan, "Segmentation-based JPEG image artifacts reduction," U.S. Patent No. 5,495,538.
11. E. R. Dougherty, *An Introduction to Morphological Image Processing*, Vol. TT9, SPIE, Bellingham, WA (1992).
12. L. Vincent, "Morphological algorithms," in *Mathematical Morphology in Image Processing*, E. R. Dougherty, Ed., Marcel–Dekker, New York (1992), pp. 255–288.
13. L. Vincent, "Morphological transformations of binary images with arbitrary structuring elements," *Signal Process.* **22**, 3–23 (1991).

Ricardo L. de Queiroz received his BS degree from Universidade de Brasilia, Brazil, in 1987, his MS degree from Universidade Estadual de Campinas, Brazil, in 1990, and his PhD degree from University of Texas at Arlington, in 1994, all in electrical engineering. During 1990–1991 he was a research associate with the DSP research group at Universidade de Brasilia. In 1993 he received the Academic Excellence Award from the Electrical Engineering Department of the University of Texas at Arlington and in 1994 he was a teaching assistant at the same university. He joined Xerox Corp. in August 1994, where he is currently a member of the research staff at the Color and Digital Imaging Systems Lab. group. His research interests are multirate signal processing and filter banks, image and signal compression, color imaging, and processing of compressed images.

Reiner Eschbach received his MS and PhD in physics from the University of Essen in 1983 and 1986, respectively. He joined Xerox in 1988 where he became a Principal Scientist at the Xerox Digital Imaging Technology Center in 1994. His research interests include color image processing, digital halftoning, and compression. He is the editor of the Recent Progress Series of IS&T.