SEGMENTATION OF COMPRESSED DOCUMENTS

Ricardo L. de Queiroz and Reiner Eschbach

Xerox Corporation
800 Phillips Rd, M/S 128-27E, Webster, NY 14580
queiroz@wrc.xerox.com and Reiner_Eschbach@wb.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 desired region can be identified and cropped (or replaced) from the compressed data without decompressing the image.

1. INTRODUCTION

In order to save storage space, compression became an important part of scanning and printing systems. The JPEG (Joint Picture Expert Group) 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 [1]. JPEG has several modes of operation for different applications. For simplicity, we discuss the most popular mode, known as baseline JPEG [1], concentrate on the luminance component of color images, and ignore header formats, restart markers and byte stuffing procedures [1]. We employ the term JPEG to designate baseline JPEG, unless otherwise stated. Due to space limitations we avoid discussing JPEG and refer the reader to [1]. A companion paper [2] also discusses image processing operations in JPEG-compressed domain, while this paper is presented in more detail in [3].

In most cases, the image is only available in compressed format and it is desirable to quickly indentify halftones, continuous tones, text, graphics, background, etc. Decompression, followed by segmentation and recompression would be very expensive in terms of storage memory and processor usage. The objective of this paper 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. The specific segmentation algorithm is a simple example intended to support the proposed framework for segmentation in compressed domain and can be replaced by a more sophisticated approach.

We use a test document shown in Fig. 2, which has resolution of 2424×2824 pixels (almost 6×7 inches at 400 ppi), and was compressed at a compression ratio around 10:1. In the discussion, it is assumed that documents are scanned at a reasonably high resolution (for good image quality).

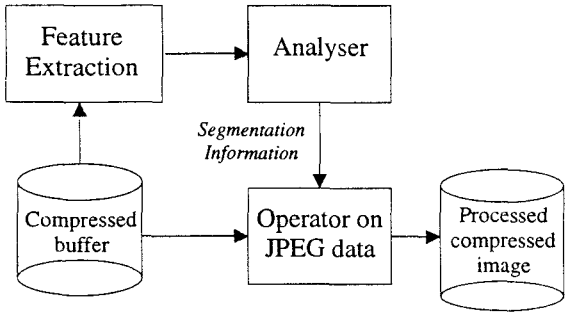

Figure 1: Objective of the paper: a system that can easily extract information from a JPEG image, analyse the information and process the compressed stream, without decompressing the image.

We now review representative functions which are helpful to enable segmentation. For example, 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 and is fed to a segmentation algorithm which uses it as an *original* image. This will result in a small image, but resolution is compromised. An alternative 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

$$E_{ac} = \sum_{ij,ij\neq 00} c_{ij}^2 \quad \text{or} \quad E_{ac} = \sum_{ij,ij\neq 00} |c_{ij}| \qquad (1)$$

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.

2. COST MAP

We propose to use another measure of activity for an 8×8 block as an input for the segmentation process. We define the encoding cost map (ECM) as the number of bits spent to encode each block In this case, each block of the compressed image maps to an integer number constructing a 2D map (image). For block coordinates (i, j) we denote the ECM entry as b_{ij}. The ECM corresponding to the test image is shown in Fig. 3. 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.

0-8186-8183-7/97 $10.00 © 1997 IEEE

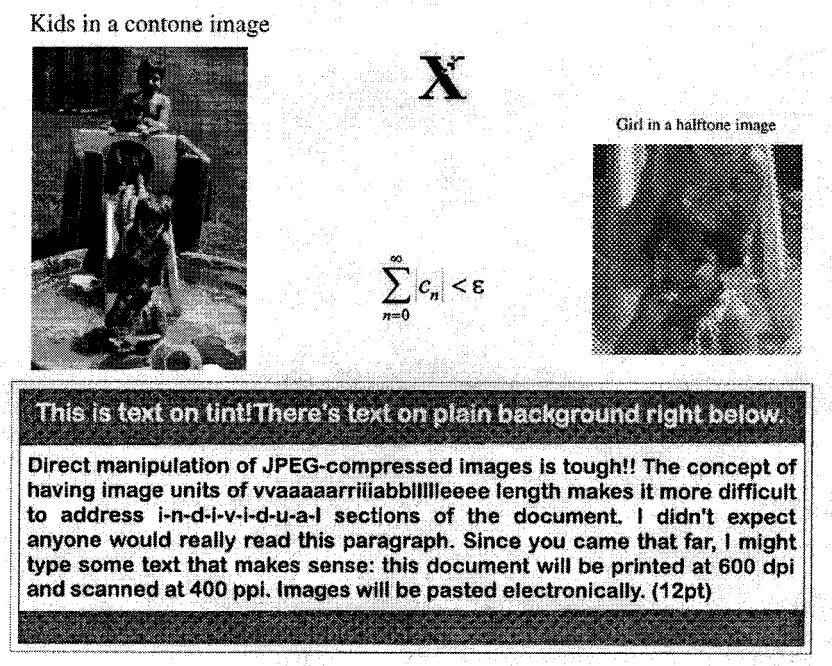

Figure 2: Test image (shown subsampled) after decompression at a ratio of 10.68:1.

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, since the decoder does not need to reconstruct coefficient sign, offset, etc. Furthermore, JPEG provides compression for images which fit a model of smooth image regions. Smooth blocks tend to generate low-frequency DCT coefficients and few bits, while active blocks tend to generate high-frequency coefficients, too, and more bits. For example, edges and texture might produce a large quantity of encoding bits.

Another important functional property of the ECM is the fact that it can be used to address individual image blocks inside the compressed data [2]. 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 [2]. We will discuss this topic later on.

The ECM general aspect is reasonably independent of certain compression parameters such as the compression ratio. 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 of $CR = bN_1N_2/\sum b_{ij}$. If B is the average ECM entry (i.e. $B = \sum b_{ij}/(N_1N_2)$), then $CR = b/B$. As we change the compression ratio, the ECM changes. For a scalable algorithm we expect that as we decrease the compression ratio, the bits are proportionally distributed to all blocks, or at least to individals regions. 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

$$B_R = \frac{1}{N_R} \sum_{ij \in R} b_{ij}. \tag{2}$$

Tests on target documents show that for most compression ratios of interest,

$$\frac{B_R}{B} \approx constant. \tag{3}$$

Therefore, by setting relative levels in a segmentation algorithm, one can scale these levels according to the compression ratio obtained.

3. SEGMENTATION OF THE ECM

Various segmentation and classification approaches have been published and most of them are based on examining an uncompressed image and extracting high-frequency information (edges) to localize letters, graphics, etc. See [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 [2] no extra computation is necessary.

The amplitude of the ECM entries for the different image regions overlap and histogram analysis of the ECM may not suffice for any robust segmentation. We would like to identify the following regions: halftone or tint, contone, text or

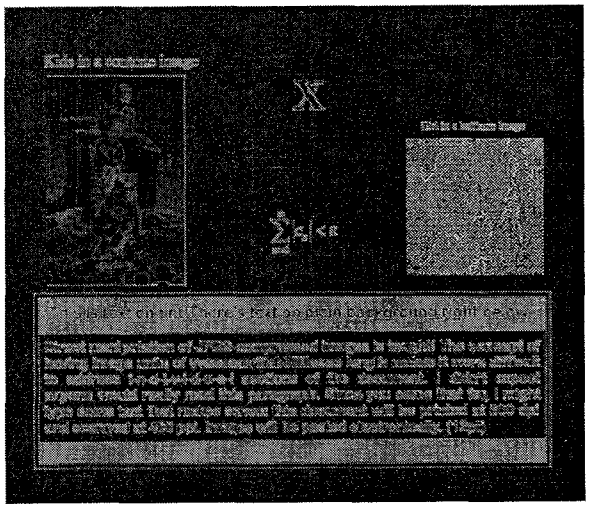

Figure 3: ECM for the test image.

graphics, and background. Each one has its own characteristics in terms of amplitude and spatial distribution of ECM pixels. Next, an example ECM segmentation algorithm is described. merely serving as an example to support the use of ECM for segmentation.

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 filters and binary morphological operations [11],[12].

Halftone - Regions with very dense concentration of bright pixels in the ECM may indicate a halftone. Text and graphics regions produce a more sparse distribution of bright ECM pixels because there is background between the letters. We filter the ECM using an averaging filter of dimensions $n_0 \times n_0$ and threshold the result (threshold t_0) yielding a mask with the position of the ECM pixels above the threshold value. From this starting mask we have to separate halftone from text and contone. The assumption is that 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 is repeated with a larger structuring element ($m_1 \times m_1$) so that text letters are 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.

Background - The background (paper) is a region where the predominant pattern is composed of dark ECM pixels, among few lighter ones. We filter the ECM using an $n_1 \times n_1$ averaging filter and threshold the result, so that values below 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, i.e. to select the blocks where the filtered ECM is below the threshold t_1 and the filtered DC (using same filter) is above threshold t_2,

Contone - We start with the mask marking the ECM pixels left after suppression of halftone and background to be separated into contone and text regions. We use morphological opening with a structuring element of size $m_4 \times m_4$ which should be larger than the text letters to 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.

Text and Graphics - After suppressing the contone mask, whatever is left is considered to be text and graphics.

The segmentation algorithm was formulated as a function of several parameters which must be adjusted. 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 setup the rest of the parameters we have to calculate the following: (a) average bit-rate (B) in bpp; (b) maximum letter size S_L of typical text (in blocks); (c) regional average bit-rate (B_R/B); (d) average paper brightness intensity in background regions (I_B); (e) minimum background brightness intensity value I_{min}, so that 95% of the background image pixels are above I_{min}.

For reasons explained in [3] we set the thresholds as:

$$t_0 = 64\, B\, \frac{1}{2}\left(\frac{B_{halftone}}{B} + \frac{B_{text}}{B}\right) \quad (4)$$

$$t_1 = 64\, B\, \frac{1}{2}\left(\frac{B_{background}}{B} + \frac{B_{text}}{B}\right) \quad (5)$$

$$t_2 = I_{min}. \quad (6)$$

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 chose m_2 as the largest integer number smaller than S_L. The final step in finding halftones is a mask dilation which can use $m_3 = 3$, since this step is just a precaution. m_4 follows the same reasoning used to chose 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.

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 histograms in [3] we set B_R/B to be 2.3, 1.25, and 0.5 for halftone, text and background regions, respectively, so that we can set the remaining parameters. The resulting masks for the test image are shown in Fig. 4.

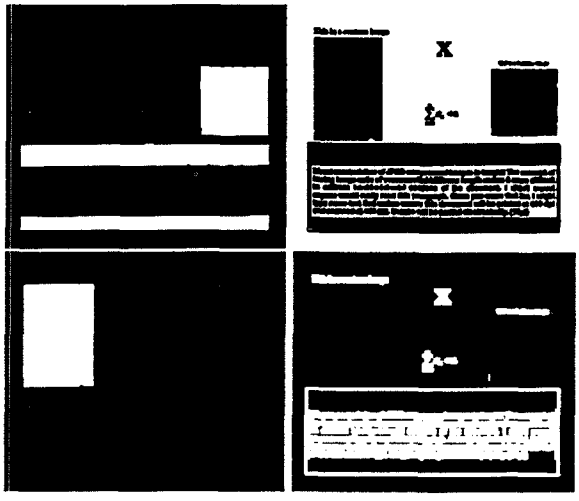

Figure 4: Segmentation masks for the test image: (a) halftone; (b) background; (c) contone; (d) text and graphics.

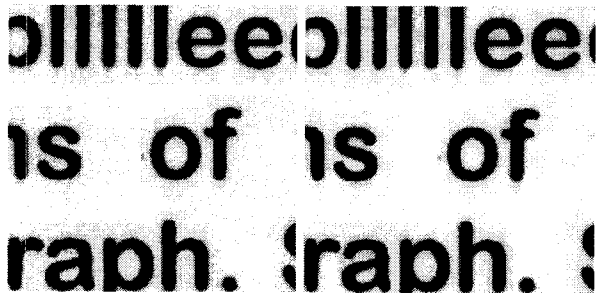

Figure 5: Stitching problems at background seam: result using a σ-filter to process pixels in the boundary region.

4. ON EXTRACTING SELECTED REGIONS

Once the ECM is segmented by binary masks we can extract the necessary information from the compressed bit-stream. The ECM and the DC map provide means to access individual blocks in the image without decompressing the bit stream. More details of that operation can be found in [2]. As we know the size of each block and the real DC component, we can quickly *seek* the file to the start of the n-th block and grab the exact amount of bits pertaining to the n-th block. We can either crop a desired region or replace it by a fixed luminance value I_B. The replacement is also very easily accomplished in the JPEG domain by merely replacing the block data by a fixed stream of bits meaning "no DC difference" and "end-of-block". In region replacement there might be stiching problems. This occurs because the paper surface is not perfectly smooth creating seams between the regions with artificial and natural background. However, these artifacts can be easily removed by applying a σ-filter only at the region boundaries (see [3] for details. Fig. 5 shows a portion of our image where the natural background was replaced by artificial background. The same figure shows the result after postprocessing.

5. REMARKS

The algorithm as described in this paper 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, as shown by our successful tests on several images.

The key of 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. Furthermore, the mask enables cropping and pasting in the JPEG domain, while a space-domain algorithm would have to require full JPEG recompression.

As a note, the algorithm will no longer work for low resolution documents, where details of text letters will have size comparable to the block size.

6. REFERENCES

[1] W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Compression Standard,"* New York, NY: Van Nostrand Reinhold, 1993.

[2] R. de Queiroz, "Processing JPEG-Compressed Images and Documents," in this Proceedings.

[3] R. de Queiroz and R. Eschbach, "Fast segmentation of JPEG-compressed documents," to appear in J. of Electronic Imaging.

[4] U. S. Patent 5,521,718, *Efficient iterative decompression of standard ADCT compressed images,* R. Eschbach.

[5] H. T. Fung and K. J. Parker, "Segmentation of scanned documents for efficient compression," *Proc. SPIE: VCIP,* Orlando, FL, Vol. 2727, pp. 701–712, 1996.

[6] S. N. Srihari, "Document image understanding," *Proc. ISCAS,* pp. 87–96, 1986.

[7] T. Pavlidis and J. Zhou, "Page segmentation and classification," *CVGIP: Graphical Models and Image Proc.,* Vol. 54, pp. 484–496, Nov. 1992.

[8] D. Dunn, T. Weldon, and W. Higgins, "Extracting halftones from printed documents using texture analysis," *Proc. ICIP,* Lausanne, CH, Vol. II, pp. 225–228, 1996.

[9] U. S. Patent 5,535,013, *Image data compression and expansion apparatus, and image area discrimination apparatus,* K. Murata.

[10] U. S. Patent 5,495,538, *Segmentation-based JPEG image artifacts reduction,* Z. Fan.

[11] E. R. Dougherty, *An Introduction to Morphological Image Processing,* SPIE Vol. TT9, Bellinghan, WA: SPIE Press, 1992.

[12] L. Vincent, "Morphological algorithms," in *Mathematical Morphology in Image Processing,* E. R. Dougherty ed., pp. 255–288, New York, NY: Marcel-Dekker, 1992.