

# Low Complexity Guaranteed Fit Compound Document Compression

Debargha Mukherjee, Christos Chrysafis, Amir Said

Compression and Multimedia Technologies Group  
Hewlett Packard Laboratories  
Palo Alto, CA 94304

## ABSTRACT

We propose a new, very low complexity, single-pass, algorithm for compression of continuous tone compound documents, known as GRAFIT (**GuaR**Anteed **FIT**) that can guarantee a minimum compression ratio of as much as 12:1 and even more, for all images in a single pass, while maintaining visually lossless quality when reproduced at resolution 300 dpi or more. The compression ratio is guaranteed in a single pass irrespective of the image being compressed. The complexity of the proposed encoder and decoder is orders of magnitude smaller than all image compression algorithms known today. For electronic compound documents, text is always compressed lossless, and depending on the type of image, the actual compression ratio achieved may be as high as 200:1 or more. For photographic images, while the compression performance is inferior to DCT or wavelet coders, for documents at resolution 300 dpi and above, the quality is still visually lossless. Overall, performance of GRAFIT is highly competitive with the more expensive algorithms like JPEG2000, JPEG, or JPEG-LS and this performance is achieved in a single pass at much lower cost in both software and hardware.

## 1. INTRODUCTION

The evolutionary path of image compression, from DCTs with Huffman coding to sophisticated wavelet transforms with arithmetic coding, has been dominated so far by the need to represent natural photographic images in a compact manner. However in recent years, with the rapid encroachment of digital images to areas like printing and commercial publishing, coupled with the need for fidelity of representation across platforms with varying rendering abilities, has resulted in new requirements for image compression that go beyond natural smoothly varying images. In particular, today's softwares and hardwares need to deal with rasterized images referred to as compound documents that combine an arbitrary mix of text, graphics and photographic images. Because traditional DCT or wavelet-based codecs do not provide efficient compression on text and graphics, new algorithms that adapt to content automatically have to be designed. In addition, because these rasterized images are often at a much higher resolution than traditional images, encoding and decoding complexity and run-time memory requirements come to be of vital consideration. Finally, the need for high quality reproduction makes low bit rate performance inconsequential. Instead, it is more important to be able to fit an image in pre-allocated buffer after compression while guaranteeing visually lossless reconstruction quality. The compactness achieved beyond that is immaterial.

Based on the above considerations, our motivation in this

work has been to develop a new methodology for high-resolution compound image compression, delivering the following features in combination:

1. *Content adaptive high quality compression*: Visually lossless quality on text, graphics and photographic images alike. The quality we are aiming at is much higher than what is achieved by Internet-oriented layered document compression schemes like DjVu [1] and others [2].
2. *Very low complexity*: Complexity significantly lower than any transform based coding scheme for both encoding and decoding. Low run-time memory requirements. Decoding needs to be especially low complexity. Both encoding and decoding complexities need to be several times lower than that achieved by DCT or wavelets.
3. *Single Pass Compression and Decompression*. The image needs to pass through the encoder in a single pass, as also the compressed bit-stream through the decoder.
4. *Guaranteed-fit operation*. The encoder takes a parameter  $c$  specifying the minimum desired compression ratio. The compressed bit-stream is guaranteed to provide at least  $c$ :1 compression as long as  $c$  is less than a certain high limit. The limit is typically 15 but depends on the implementation. Note that running JPEG on an image with a given quality factor cannot guarantee a compression ratio.

A review of the existing image and compound document compression algorithms in vogue revealed that none of them support all the above requirements. It was this unmet need that motivated the development of GRAFIT, an acronym for **GuaR**Anteed **FIT** compression.

## 2. RATE-CONTROLLED MULTIMODE CODING

The GRAFIT algorithm is block-based. Each image is made up of 8-line strips, and each strip is further broken into 8×8-pixel blocks, which is the elemental processing cum coding unit. All input images are considered to be in true-color RGB with 24 bits per pixel. Each 8×8 block is described and coded by one of several possible *modes*. A particular implementation of GRAFIT may choose to implement only a subset of all the supported modes, and use its own mode selection rules, but a full functional GRAFIT decoder is able to decode any version of GRAFIT, irrespective of the modes and mode selection procedure used during encoding.

In order to guarantee a minimum desired compression ratio in a single pass, the supported modes in a GRAFIT encoder are wrapped within a *greedy* multimode rate-control framework, as shown in Figure 1. A GRAFIT encoder is run on an image with

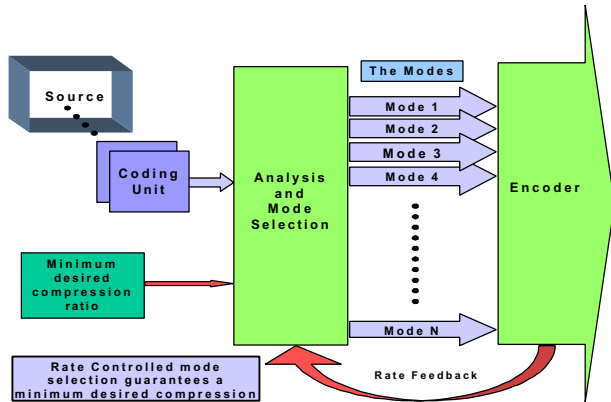


Figure 1. Rate-controlled multimode coding

a parameter  $c$  denoting the minimum desired compression ratio. This yields a base bytes value  $B_b = 192/c$  for the number of bytes available on an average to encode each  $8 \times 8$  RGB block. The set of modes that always require description lengths lower than the average available bytes figure  $B_b$  are called *base modes*. The base modes can always be selected for any block in an image. The remaining modes yield description lengths higher than  $B_b$ , and are referred to as *enhancement modes*. Enhancement modes are selected only when bytes have been saved previously to afford using more bytes, but also when it is necessary to avoid loss of visual quality.

In particular, for each input image block, the encoder computes the maximum number of bytes available to encode it, denoted  $A_b$ . This number is obtained by adding a bytes saved so far value, denoted  $S_b$  (initialized to 0), to the base available bytes figure  $B_b$ . Of all the supported encoding modes, some may require longer descriptions and some shorter descriptions than  $A_b$ . The mode selection process restricts the choice of encoding modes to only the subset of modes that yield lengths lower than  $A_b$ . Of this reduced subset, the mode that yields the best representation for the given image block is picked. Once the block is coded, the actual number of bytes used for the block, denoted  $L_b$ , is fed back to the encoder. The saved bytes value  $S_b$  is updated to  $A_b - L_b$  for use in computation of the available bytes figure for the next block. In this manner, the compression ratio always remains higher than the minimum desired compression ratio, and the number of available bytes for each block remain at least equal to  $B_b$ .

### 3. GRAFIT MODES

In order to guarantee low complexity, it is imperative that all the possible modes in GRAFIT be very simple. While each mode is quite primitive by itself, when used in conjunction with a large number of other simple modes in an adaptive multimode framework, a very powerful coding scheme is obtained.

The most compelling GRAFIT modes are related to Block Truncation Coding (BTC) [3], [4], and Vector Quantization (VQ) [7]. Block Truncation Coding (BTC), first proposed by Delp and Mitchell [3] in 1979, refers to a family of coding techniques where small blocks of monochrome images are represented with two grayscale levels 8 bits each, and a 1-bit/pixel binary mask denoting the level to use for each pixel.

While the coding scheme is lossy in principle, it is to be noted that for blocks with only two levels – as is commonly encountered in text and graphics portions of an image – the scheme becomes lossless. A comprehensive survey of BTC and its variants has been presented by Franti *et al* [4].

Vector Quantization (VQ) [7] refers to a popular signal compression scheme where multiple scalar elements are grouped together and reproduced using a few vector reproduction levels. For RGB color images, 2-level BTC may be readily combined with color VQ to yield BTC-VQ [5], [6]. In BTC-VQ, small blocks of color images are represented with two RGB reproduction levels 24 bits each, and a 1-bit/pixel binary mask. Thus, a compression ratio of 6:1 is immediately achieved for  $4 \times 4$  blocks, and 13.7:1 for  $8 \times 8$  blocks. The 2-level VQ methodology can be readily extended to allow more than two reproduction levels. In GRAFIT, the number of possible reproduction levels can be up to eight. When a block is represented using a  $n$ -level VQ ( $n \leq 8$ ) mode,  $n$  RGB colors (the VQ codebook) as well as a  $n$ -ary mask (coded indices) for the pixels in the  $8 \times 8$  block are transmitted in the bit-stream. Although BTC-VQ is a special case of  $n$ -level VQ with  $n = 2$ , in GRAFIT, two-level BTC-VQ gets special attention in being applicable to  $4 \times 4$ ,  $4 \times 8$ , or  $8 \times 4$  subblocks of a  $8 \times 8$  input block.

Besides 2-level BTC-VQ and  $n$ -level VQ modes, blocks that show smooth gradients are described using the interpolated mode, where a  $8 \times 8$  block is downsampled to  $2 \times 2$  and transmitted with the bit-stream.

Blocks that cannot be described efficiently by any of the modes above, and there are enough bytes available to afford it, use raw or truncated raw encoding. A raw block is transmitted as is in the bit-stream with no compression, while a truncated raw block is transmitted after truncating several least significant bits from each color component.

More details about the four mode families and how they work are presented below, and in Figure 2:

1. *n-level VQ modes*, where  $n = 1, 2, \dots, 8$ : In these modes, the block is represented using exactly  $n$  colors or reproduction levels. These modes are used primarily to represent text and graphics blocks. The representation for these modes can either be lossy or lossless. The encoder transmits  $n$  color values and an  $n$ -ary mask to the decoder to denote which pixel is to be reproduced using which color. The color values are represented using either 24 bits or 12 bits, depending on the available bytes figure for the current block. When 24 bit color representations are used, an *adaptive dictionary* of colors is consulted in order to represent some of the colors in an economical manner. Only colors that do not already occur in the dictionary need to be transmitted, and the rest are indexed from the dictionary. An  $n$ -ary mask needs to be transmitted in addition to the  $n$  colors, whenever  $n > 1$ , in the  $n$ -color modes. No mask is needed for single color blocks ( $n = 1$ ). Instead, a special *run-length mode* is used to represent compactly a set of consecutive single color blocks whenever encountered. This is extremely efficient for images with white constant background as is common in compound documents. When  $n = 2$ , a 1-bit/pixel mask is

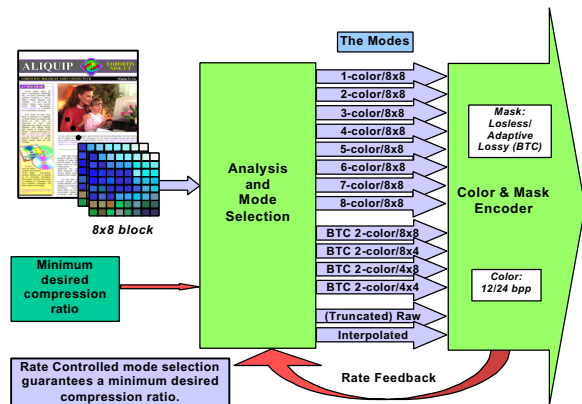


Figure 2. GRAFIT modes

conveyed. For  $n = 3$ , a set of five ternary mask symbols, which occur in a ring of size  $3^5 = 243$ , are represented at once using one byte of mask data. For  $n = 4$ , a 2-bit/pixel mask is used. For  $n = 5$  or 6, a set of three symbols, which occur within a ring of size  $6^3 = 216$  are represented with one byte of mask data. Finally, for  $n = 7$  or 8, a 3-bit/pixel mask is used.

2. **2-level BTC-VQ modes:** In these modes, a block is coded using 2-level BTC-VQ over a region of support, that may either be the entire  $8 \times 8$  block, or two  $8 \times 4$  blocks, or two  $4 \times 8$  blocks, or four  $4 \times 4$  blocks obtained by splitting the  $8 \times 8$  block. The encoder forces 2 colors per  $8 \times 8$ ,  $8 \times 4$ ,  $4 \times 8$ , or  $4 \times 4$  block. The two colors per support block, and a binary mask for the entire  $8 \times 8$  block is transmitted to the decoder. The color values are represented using either 24 bits or 12 bits, depending on the available bytes figure for the current block. These modes are used primarily to represent photographic images in a lossy manner. Normally, the mask is conveyed at 1 bit/pixel; but there is a mode where *adaptive lossy compression* of the mask is used if the two reproduction levels in a support block are very close.
3. **Interpolated Modes:** In these modes, the color components of an  $8 \times 8$  image block are downsampled by a factor of 4 in both directions, and the downsampled colors are transmitted at 12 or 24 bits/pixel. The decoder reconstructs the block by smooth interpolation. These modes are effective for coding blocks where color changes in a gradual manner.
4. **Raw and Truncated Raw modes:** In these modes, each pixel is represented with either 24 bits/pixel, or 12 bits/pixel, or 8 bits/pixel. Only the 24 bits/pixel raw mode is lossless, while the truncated raw (12 and 8 bits/pixel) modes are lossy.

#### 4. MODE SELECTION AND ENCODING

While the GRAFIT framework allows for a wide range of modes in which a single  $8 \times 8$  block may be encoded, a particular implementation of GRAFIT may use only a subset of them. Depending on the system requirements, designers must trade off complexity and performance carefully, to arrive at a GRAFIT codec implementation supporting a specific subset of modes,

called a GRAFIT *profile*. Furthermore, for each profile, a variety of GRAFIT encoders may be developed achieving various trade-offs between complexity and performance. The challenge is to develop a systematic block analysis procedure at the encoder, so that an appropriate mode selection based on block characteristics and available bytes  $A_b$  is made, while keeping encoding complexity low and reproduction quality high.

In this paper we describe a GRAFIT profile and an associated encoder where all the modes described in Section 3 are used. This profile is designed to achieve minimum compression ratios as high as 12:1 at visually lossless quality if reproduced at 300 dpi or higher resolution. The encoder flowchart is shown in Figure 4. In this diagram, the blocks marked with a red circle denote blocks where rate feedback information is utilized. Decisions based on available bytes are made at almost every stage, in order to save bytes.

For each new block, the first step is to determine the maximum number of colors,  $m$  that can be transmitted in the  $n$ -color mode, with the current available bytes. This figure is obtained based on the number of bytes required to transmit  $m$  colors in  $m$ -color mode with 12-bit colors. Once  $m$  is obtained, a color counting routine checks to see if there are  $m$  or fewer distinct colors in the block using exact 24-bit color matches. In particular, pixels are scanned sequentially, and are either found to be exactly equal in color to one of the previously encountered colors, or used to create new colors. If the test succeeds, the colors obtained are transmitted in  $n$ -color mode. The dictionary is consulted initially to check if any of the distinct colors already exist in it. Eventually however, depending on the current available bytes and number of dictionary matches found, either 24-bit colors with dictionary, or 12-bit colors without dictionary are transmitted.

If the test for distinct colors fail, the next step is to compute the maximum color range of the R, G, and B components in the  $8 \times 8$  block, given by  $\max[R_{\max} - R_{\min}, G_{\max} - G_{\min}, B_{\max} - B_{\min}]$ . If this quantity is above a threshold for high contrast, denoted  $T_{HC}$ , then control passes to a color clustering routine that looks for up to  $m$  distinct color clusters. The routine uses a single pass clustering technique, where pixels are scanned sequentially, and are either included in one of the previous clusters if close enough, or used to create new clusters if significantly different from all previously created clusters. Note that this is a very simple and fast method for VQ design albeit sub-optimal. The complexity requirements are too stringent to support LBG VQ design [8]. If  $m$  or fewer clusters are obtained, the block is represented in  $n$ -color mode with the representative colors being the average or the mid-point of the respective clusters. For color encoding, as in the case for distinct colors, either 24-bit colors with dictionary, or 12-bit colors without dictionary are used, based on the available bytes figure. Note that it is this clustering stage that incorporates robustness to scanning noise in the algorithm.

If the test for  $m$  clusters fails, the encoder computes the maximum of the horizontal and vertical gradients in the block. The maximum gradient value is compared against a threshold  $T_{GR}$ , and if found to be less, the block is encoded in the Interpolated mode by downsampling it by a factor  $4 \times 4$  by

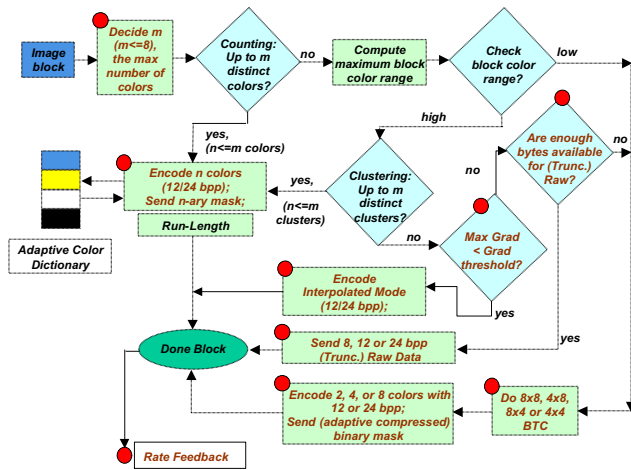


Figure 4. GRAFIT encoder flowchart

averaging, and transmitting the four downsampled pixels in the bit-stream using 12- or 24-bit colors.

If the gradient test fails the encoder checks to see if there are enough bytes available to represent the block in 8 or 12 bpp truncated raw modes, or in the 24 bpp raw mode. If so, the appropriate number of bytes is transmitted to the bit-stream.

If there are not enough bytes available for raw or truncated raw, or if the maximum color range computed before clustering is lower than the  $T_{HC}$  threshold, then the block is represented using one of the BTC-VQ modes. The decision as to which support block size ( $4 \times 4$ ,  $4 \times 8$ ,  $8 \times 4$ , or  $8 \times 8$ ) will be used, is determined partly by the available bytes figure, and partly by the maximum color ranges (defined earlier) for each  $4 \times 4$ ,  $4 \times 8$ ,  $8 \times 4$  or  $8 \times 8$  support block. The idea is to choose support blocks of as small color ranges as possible, without exceeding the available bytes figure. Depending on the support block size chosen, the required number of colors is transmitted with 24 or 12 bits each. Additionally, adaptive lossy compression of the mask is turned on, if at least one pair of colors is found to be close to each other. A simplified form of LBG design procedure [8] is used to determine the dual color quantization levels in the BTC-VQ modes.

## 5. RESULTS

The GRAFIT algorithm is tested by applying it to a test suite of ninety 300 and 600 dpi images, and comparing its performance in Figure 3 with JPEG at quality factor 75, and TAOS – a fast, lossless, line-based algorithm. The test suite contains images of widely varying types, ranging from entirely photographic to entirely text/graphics but mostly containing arbitrary mixes of both. In order to compare the performance profile for the entire test suite rather than individual images, each curve is sorted independently so that the data corresponding to the same sorted image number in the x-axis in each plot actually corresponds to different images. The version of GRAFIT in these plots was run with the minimum compression ratio parameter 12:1, while the JPEG quality factor was 75.

The compression ratio plot shows that the compression achieved for all images with GRAFIT is at least 12:1. While it

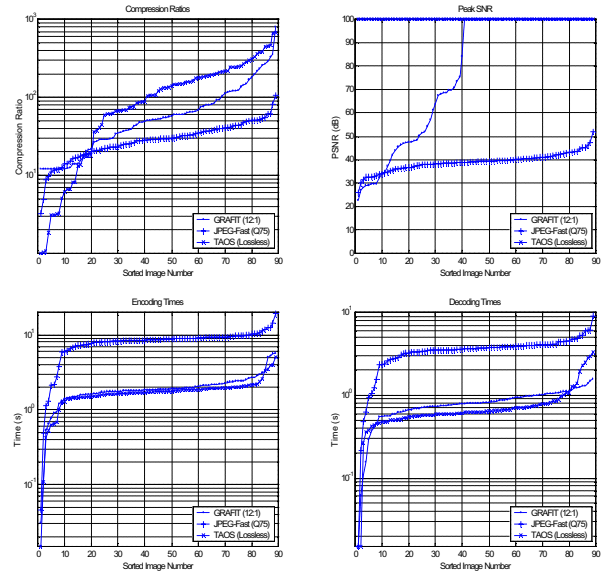


Figure 3. Results comparing GRAFIT (minimum compression ratio 12:1), JPEG, and TAOS (Lossless line-based algorithm) for a test suite of 90 compound images.

is not surprising that this can never be achieved by a lossless scheme, it is also not achieved by JPEG for certain hard to compress images in a single pass. The PSNR plot shows that with GRAFIT, a vast majority of compound documents are actually coded losslessly (showed as PSNR 100 in the plot). These images are actually electronically generated ones and contain mostly text and graphics. It is only for certain images with a significant photographic component that JPEG beats GRAFIT. Finally, the encoding and decoding times show GRAFIT as several times faster than JPEG but equivalent to TAOS.

## 6. REFERENCES

- [1] P. Haffner, L. Bottou, P.G. Howard, P. Simard, Y. Bengio, Y. LeCun, "High Quality document image compression with DjVu," *Journal of Electronic Imaging*, pp. 410-25, July 1998.
- [2] D. Mukherjee, N. Memon, A. Said, "JPEG-Matched MRC compression of compound documents," *Proc. IEEE Int. Conf. Image Processing*, Thessaloniki, Greece, Oct 2001.
- [3] E. J. Delp and O. R. Mitchell, "Image compression using block truncation coding," *IEEE Trans. Comm.*, vol. COM-27, pp. 1335-42, 1979.
- [4] P. Franti, O. Nevalainen, T. Kaukoranta, "Compression of digital Images by block truncation coding: a survey," *Computer Journal*, col. 37, no. 4, pp. 308-32, 1994.
- [5] Y. Wu and D. C. Coll, "Single bit-map block truncation coding of color images," *IEEE J. Selected Areas in Comm.*, vol. 10, pp. 952-959.
- [6] T. Kurita and N. Otsu, "A method for block truncation coding for color image compression," *IEEE trans. Commun.*, vol. 41, no. 9, pp. 1270-74, Sept. 1993.
- [7] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Boston, MA: Kluwer, 1992.
- [8] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, no.1, pp. 84-95, Jan. 1980.