TROBIC: TWO-ROW BUFFER IMAGE COMPRESSION

Viresh Ratnakar

Epson Palo Alto Laboratory
3145 Porter Drive, Suite 104
Palo Alto, CA 94304

ABSTRACT

We describe a color image compression and decompression scheme suitable for high resolution printers. The proposed scheme requires only two image rows in memory at any time, and hence is suitable for low-cost, high-resolution printing systems. The compression ratio can be specified and is achieved exactly. Compound document images consisting of continuous-tone, natural regions mixed with synthetic graphics or text are handled with uniformly high quality. While the target compression ratios are moderate, the quality requirements are extremely high: the compressed and decompressed printed image needs to be virtually indistinguishable from the original printed image. The scheme combines a lossless block coding technique with a wavelet block codec. The wavelet block codec uses a new and simple entropy coding technique that is more suitable for the specific block-structure, compression target, and discrete wavelet transform used.

1. INTRODUCTION

Printer technology today can support very high resolution printing: resolutions as high as *1440×720* dots per inch (dpi) are becoming increasingly available. At these resolutions, the size of a single digital image is of the order of tens or hundreds of megabytes. For example, an *11"×8.5"* color image at *1440×720* dpi requires *15840×6120×3* bytes (each of the three color channels, R, G, and B, uses one byte per pixel), i.e., *290* megabytes. Compression becomes a very important requirement for such systems, in order to reduce printing time as well as printer cost. Compression can reduce printing time by speeding up the data transfer to the printer. A more serious requirement arises inside the printer; often, print-engines require the entire image to be available in memory before they can start printing. In fact, print-engine speeds in terms of pages-per-minute are usually specified assuming that all the page images are in memory. Low-cost printers may not be able to afford to carry *290* megabytes (or more) of RAM. This problem can be alleviated by using compression and inserting a software/hardware decompression module between the printer memory and the print engine.

The proposed scheme, Trobic (Two-Row Buffer Image Compression) seeks to provide a low-cost solution to the printer compression problem. A major constraint for designing Trobic was that its memory requirement be minimal, to enable inexpensive modules that can be easily plugged in at any point during the printing pipeline. Trobic uses only two image rows at any time. An additional crucial requirement was that at *3:1* compression the quality of the printed image should not suffer at all, and that quality of compound images consisting of text, line-art, graphics, and natural imagery should be uniformly high. While images displayed on computer monitors may accommodate substantial distortion before the eye can notice them, for a high quality printed image, a single off-color speck on a uniform area becomes unacceptably objectionable. Finally, Trobic had to be designed to achieve guaranteed compression, without which the printer would be unable to print images whose compressed size exceeds the available memory.

2. OVERVIEW

In order to handle compound images well, Trobic uses a block-based coding approach, in which each block is automatically classified as natural or non-natural and is coded accordingly. After extensive experiments, the block dimensions were chosen as *32×2* (block height is *2*, to meet the two-row buffer requirement). Let W and H denote the width and height (in pixels), respectively, of the input color image, and assume (for simplicity) that W is an exact multiple of *32* and H is even. Let ρ denote the target compression factor (that is, the size of the compressed image should not exceed $W \times H \times 3 \times 8 \times \rho$ bits). The Trobic encoder processes the image block by block, using the following strategy:

- ***Bit-budget apportionment:*** The overall bit budget is dynamically distributed to the blocks such that each block is guaranteed to get a bit-budget of at least $32 \times 2 \times 3 \times 8 \times \rho$ ($= 1536\rho$) bits. In any of the block coding modes described below, the specified block bit-budget is guaranteed not to be exceeded. Some bits may be saved while coding a block within its budget. While it is possible to have a complex bit-budget apportionment strategy that distributes the budget according to block complexity and distributes the saved bits too according to some sophisticated criteria, it is difficult to absolutely guarantee compression as well as quality using such schemes. For example, if the overall budget is distributed such that simple blocks get less than their uniform share (1536ρ bits) and complex blocks are allocated more than their uniform share, image quality will suffer near the end of the image if it consists of lots of complex blocks. Our simple apportionment strategy can be described as follows: if n is the number of blocks coded previously, and B_n is the number of bits used thus far, then the next block is allocated a budget of $(1536\rho(n+1) - B_n)$ bits. Note that each block's budget is thus guaranteed to be at least 1536ρ bits.
- ***Three block coding modes:*** In order to achieve uniformly high quality in compound documents, Trobic uses three coding modes, classifying each block as a P-block, or an A-block, or a W-block. The P-block codec

is designed for synthetic regions of the image which do not have smoothly varying colors, for example, perfectly black text on perfectly white background. The P-block codec is lossless. Image blocks with smoothly varying (natural) pixel intensities get classified as W-blocks, and are coded in a lossy manner using a discrete wavelet transform. If the image consists largely of P-blocks, then the coding will be very efficient (in the sense that $(1536\rho(n+1) - B_n)$ will be very high most of the time). In this case, blocks may be coded without any compression, and are labeled as A-blocks.

- ***P-blocks:*** Depending upon the input parameter ρ, the Trobic encoder calculates a cutoff size, P (the relationship between ρ and P will become clear later). If an image block has no more than P distinct colors, then it is labeled a P-block.
- ***A-blocks:*** If an image block cannot be classified as a P-block, and if its block bit budget exceeds A (another cutoff parameter whose meaning will become clear later), and if no previous block has been labeled a W-block, then it is labeled an A-block.
- ***W-block:*** If an image block cannot be classified as a P-block or an A-block, then it is labeled a W-block.

3. P-BLOCKS

The motivation behind Trobic's block classification strategy is that a *32×2* block from a very high resolution image is not likely to have too many distinct colors if it comes from a text or line-art or graphics region in the image. These blocks (the ones with fewer than P colors) are labeled P-blocks and are coded losslessly, as any pixel distortion in such a block is likely to be very visible. When a block has several distinct colors, it can accommodate more distortion, and hence the lossy wavelet coding mode is appropriate for it (i.e., it is labeled a W-block).

The letter P in "P-block" stands for *Palette*. A P-block is compressed by first coding its palette (which is a table consisting of all the distinct colors used in the block) and then coding each pixel as an index into the palette. The cutoff parameter P is chosen such that the largest palette and the indices fit in the minimum bit budget. Thus, P is set to be the largest value that satisfies:

$$24P + 64\lceil \log_2 P \rceil \le 1536\rho - \Delta_P,$$

where $24P$ is the number of bits needed to code the palette, $64\lceil \log_2 P \rceil$ is the number of bits needed to code all the pixels as palette indices, and Δ_P is the overhead for a P-block. The overhead consists of bits needed to encode block kind, and bits needed to encode palettes size. The block kind for P-blocks is encoded in Trobic using the fixed *1*-bit code, "*0*." Trobic does not allow P to exceed *16*, and hence four bits are enough for encoding the palette size (*1-15*). Thus, $\Delta_P = 1+4 = 5$. For *3:1* compression, for example, P is obtained as *10*.

The actual palette size for a P-block may be much smaller than P. Trobic uses a novel run-length coding technique in certain situations, while still guaranteeing that the bit-budget is not exceeded. Let p denote $\lceil \log_2 P \rceil$. Let Y denote the actual palette size in a P-block, and y denote $\lceil \log_2 Y \rceil$. If y equals p, then Trobic does not use run-length coding, as in this case one cannot guarantee that run-length coding will *never* increase the coded size. If y is less than p, then after coding each pixel's index in the palette (which requires y bits), Trobic has at least one bit to spare without exceeding the budget. Trobic uses the following small variable-length code to code the length of the run of subsequent pixels with identical value:

- The code "*0*" is used to indicate a run-length of zero.
- The code "*10*" is used to indicate a run-length of one.
- The code "*11[r-2]*" is used to indicate a run-length of $r \ge 2$, where *[r-2]* denotes the binary representation of *r-2* using p bits. If r exceeds 2^p+1, then r is set to 2^p+1, and only 2^p+1 pixels are counted in the run.

It can be seen that this variable-length code ensures that the block budget is never exceeded.

4. A-BLOCKS

When a document consists mainly of text and graphics, with perhaps a small natural image or two, somewhere in the middle, Trobic is likely to have saved a lot of bits on the P-blocks, because of run-length coding. If a block cannot be classified as a P-block, but the budget available for it is very large, Trobic classifies it as an A-block and codes it "As-is," that is, it simply puts all the pixel bits from the block directly into the coded bit-stream. This requires a budget of at least *1536* bits, plus some room for the overhead for encoding block kind (which is at most *2* bits in Trobic). Thus, Trobic classifies a block as an A-block only if its budget exceeds or equals $A = 1538$ bits. Moreover, a block cannot be classified as an A-block if some previous block has been classified as a W-block. The reason is that A-blocks are very wasteful, and if some block did get classified as a W-block, it is likely that some future block will also be a W-block, and making the current block an A-block will take a substantial chunk out of the remaining budget.

Thus, during the encoding process, Trobic may classify blocks as P-blocks or A-blocks up to a point, after which the first W-block is seen. Subsequently, blocks are classified only as P-blocks or W-blocks. The code at the beginning of each block, which identifies the block kind, is set as follows: initially, "*0*" denotes a P-block, "*10*" denotes an A-block, and "*11*" denotes a W-block. If a W-block is coded, then subsequently, "*0*" denotes a P-block and "*1*" denotes a W-block.

5. W-BLOCKS

Given a W-block and a bit-budget, the Trobic encoder subtracts from the budget the number of bits that are needed to code the block kind (*2* bits for the first W-block and *1* bit thereafter). Trobic codes W-blocks in (Y,Cb,Cr) color space. The bit-budget is divided up into budgets for the Y, Cb, and Cr channels (in the proportion of 5:2:2, in the current implementation). We now describe the coding process for a single *32×2* block I of pixels, given a bit-budget, B.

This coding process is the key quality determinant in Trobic, as it is the only part that involves lossy compression. In order to provide guaranteed compression, a natural choice of method for coding these blocks is to use some energy compacting transform

and code only as much information about the coefficients as would fit in the bit-budget. Trobic uses a discrete wavelet transform (DWT) that can be described as a subband decomposition using two biorthogonal filters. The Haar filter is used on columns (the block height is only *2*), and in the last step of the subband decomposition, while in the other decomposition steps, a filter reported by Villasenor et. al. [7] is used. This filter, which we refer to as the *2-6* filter, has some nice properties for compression such as regularity, nice impulse and step response (see [7] for details), and can be implemented using integer arithmetic. We omit the filter details here, as many other suitable filters can be used to replace the 2-6 filter. For details on the implementation using the 2-6 filter, please refer to [3].

The DWT on *I* results in a *32×2* block of coefficients, *C*. We number the coefficients in raster order as *C(0)* through *C(63)*. The coefficients and their subband structure is shown, together with a *subband number* for each subband, in Figure 1.

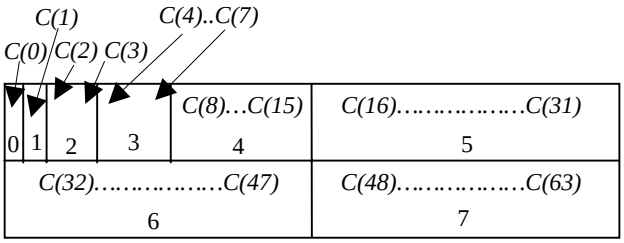

Figure 1. Trobic's DWT coefficients and their subband structure.

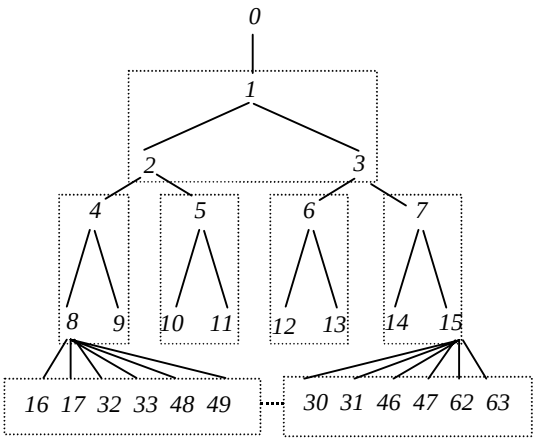

Figure 2. Trobic's DWT coefficients and their subband structure.

5.1 Tree structure of the DWT coefficients

The *64* DWT coefficients can be seen to form a tree (Figure 2). Each level in the tree corresponds to coefficients for a particular scale, while each subtree corresponds (roughly) to a subset of the original pixel area. The rectangular boxes drawn around some coefficients show the grouping used by the Trobic encoder, which will be explained in the next subsection. Level *0* of the tree corresponds to the single DC coefficient in subband *0*. Level *1* has just the one coefficient in subband *1*. Levels *2*, *3*, and *4* correspond respectively to subbands *2*, *3*, and *4*, respectively. The leaf level (level *5*) has the highest frequency coefficients from subbands *5*, *6*, and *7*. The spatial extent of coefficients from a particular subtree is seen to be the following: Each coefficient at the leaf level (*5*) corresponds to a *2×2* pixel area and each subtree rooted in level *4* corresponds to a *4×2* pixel area. Each subtree rooted in level *3* corresponds to an *8×2* pixel area and both the subtrees rooted in level *2* correspond to a *16×2* pixel area. This correspondence is only a rough approximation, as the filters used span more than *2* inputs.

In state-of-the-art wavelet-based image coders, such a tree structure is coded by utilizing the fact that very often, the absolute magnitude of a parent coefficient is no less than the absolute magnitudes of all its children [4,5]. This fact is utilized in the coding process, as follows: The absolute magnitudes of coefficients are compared against a sequence of thresholds which are successively decreasing powers of *2* (other thresholds are possible, but powers of *2* lead to simple and elegant algorithms and are most commonly used). For each threshold, some coefficients will be seen as *significant* (i.e., their absolute magnitudes will exceed or equal the threshold) for the first time. The positions and signs of all the these coefficients are coded efficiently using entropy coding of symbols denoting frequently occurring cases such as an entire subtree being zero (a *zerotree*) [4,5] and an entire subtree excepting its root being zero [4]. Once a coefficient becomes significant, its bits are progressively included in the coded bit-stream, one bit during each thresholding step. Typically, these bits are not entropy coded as they are likely to be uniformly random. The coding process continues (with smaller thresholds) until the bit-budget is exhausted. This description is meant to be a general overview; specific encoders have several details and differences.

An alternative way of looking at the above coding process (which would lead to the new coding technique used in Trobic) is as follows. Suppose we list the bits of the absolute magnitudes of the coefficients in a list, starting at *C(1)* (Figure 3).

C(1)	*0 0 0 1 1 0 1 1 0*	*0 1 0 0 0*
C(2)	*0 0 0 1 0 1 1 0 1*	*1 0 0 1 0*
C(3)	*0 0 0 1 1 0 1 0 0*	*0 1 0 1 1*
C(4)	*0 0 0 0 0 0 0 0 1*	*0 0 1 0 1*
C(5)	*0 0 0 0 0 0 0 1 0*	*1 0 1 0 1*
C(6)	*0 0 0 0 0 0 1 0 0*	*0 0 0 0 1*
C(7)	*0 0 0 0 0 0 1 0 0*	*0 0 0 1 1*
…	…	←Cutoff = *5* bits

Figure 3. Trobic's DWT coefficients and their subband structure.

Let the most significant *1*-valued bit of each coefficient be called the S-bit (the S-bits are highlighted in dark gray in Figure 3). The coefficients that are negative are identified by underlining their S-bits. Then the above coding process results in finding a cutoff

bit position (which is the smallest power of *2* used as a threshold), which is shown by a dotted line in Figure 3. The S-bit position and sign of each coefficient is encoded in the bit-stream (unless the S-bit position is to the right of the cutoff). The light-gray bits for each coefficient (bits to the left of the cutoff but excluding the S-bit) are also included in the bit-stream.

Let B_s denote the number of bits spent in coding the S-bit positions and signs, and let B_c denote the number of bits spent in coding the light-gray bits (which is the same as the number of light-gray bits, when there is no entropy coding of these bits). Then the cutoff found by the coding process has to be the right-most cutoff such that $B_s+B_c \leq B$, where B is the total bit-budget.

5.2 Tree encoding in Trobic

We found that the zerotree approach for coding DWT coefficients can be improved upon substantially, for the specific block-structure in Trobic. The reason is that at the high bit-rates and for the *32×2* block size used in Trobic, quite a few of the coefficients are non-zero. With fewer zerotrees, B_s becomes quite high as for each non-zero coefficient we have to code the exact position of its S-bit. This results in a smaller remaining budget for B_c, and hence a larger cutoff. We also tried the stack-run coding approach [6], but found the new method (explained below) to be better (again, because of the specific block-structure and the high bit-rates). This new method also has the advantage that it avoids complex entropy coding techniques such as Huffman coding and arithmetic coding, and hence is extremely simple, computationally.

Trobic groups the AC coefficients into thirteen groups, these are shown using dotted rectangular boxes in Figure 2. The idea in zerotree coding is that the S-bit position of the root of any subtree dominates that of its children. Trobic uses the fact that the S-bit position within any group is likely to be similar. For each group, the encoder codes the largest S-bit position, *s*, within the group, and then each coefficient in the group is coded as if its S-bit position were also *s*. While this results in some zero-valued bits to the left of the actual S-bits being encoded, there are substantial savings compared to the case when each S-bit position is identified exactly.

Trobic's DWT coefficient coding process can be summarized thus:

1. Put *C(0)* in the bit-stream, exactly (requires *11* bits, for Trobic's DWT).
2. Calculate the least cutoff that would allow the block to be coded without exceeding the remaining bit-budget. This can be done exactly, by keeping track of the number of coefficients whose S-bit will be to the right of the cutoff, for each possible cutoff. Put the value of this cutoff, *c*, in the bit-stream.
3. For each coefficient group, put in the bit-stream the value of the largest S-bit position, *s*, within the group. For each coefficient in the group, put *s-c* bits to the left of the cutoff into the bit-stream. If these bits are not all zero, put a bit in the bit-stream to indicate the sign.

There are some more minor issues that need to be dealt with, in order to achieve the budget exactly. The details of Trobic's W-block coding can be found in [3].

6. DISCUSSION

The ultimate objective of Trobic is that the printed image after *3:1* compression and decompression be indistinguishable from the original. This objective is achieved fairly well. The PSNR values are not particularly suitable to judge the quality of a compound image consisting of natural regions and graphics regions. The PSNR values achieved on the color Lena image are*: 50.1* dB (Y), *40.8* dB (Cb), and *40.5* dB (Cr), at *3:1* compression. For Lena, all blocks are W-blocks. We are not aware of any other lossy compression technique that uses only two image rows and guarantees to achieve the specified compression ratio, so it is hard to compare these PSNR results to anything else. It would be interesting to see how well the W-block coding technique used in Trobic can be extended to more "square" block dimensions, such as *8×8*, (the block size in JPEG [2]). While having only two image rows in memory at any time is not very efficient for natural image regions, it is actually useful in providing the granularity needed to separate graphics regions from natural regions. Further work on Trobic is focussing on ways to use Trobic's block classification and its DWT to simplify/improve some of the other steps in the printing pipeline, such as sharpening blurred images.

Please note that there is a patent application pending on the Trobic technology.

7. ACKNOWLEDGEMENT

The author thanks Bhaskaran Vasudev, Anoop Bhattacharjya, and Hakan Ancin for their valuable comments and insights.

8. REFERENCES

[1] DeVore, R. A., Jawerth, B., and Lucier, B. J. "Image Compression Through Wavelet Transform Coding." *IEEE Trans. Inform. Theory*, 38(2):719–746, March 1992.

[2] ISO 10918-1 JPEG Draft International Standard and CCITT Recommendation T.81.

[3] Ratnakar, V. "SBCW: A Simple Block Codec with Wavelets." *Epson Palo Alto Laboratory Technical Report*, November 1997.

[4] Said, A. and Pearlman, W. A. "A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees." *IEEE Transactions on Circuits and Sysems in Video Technology*, 6(3):243–250, June 1996.

[5] Shapiro, J. M. "Embedded Image Coding Using Zerotrees of Wavelet Coefficients." *IEEE Transactions on Signal Processing*, 41(12):3445–3462, December 1993.

[6] Tsai, M. J., Villasenor, J. D., and Chen, F. "Stack-Run Image Coding." *IEEE Transactions on Circuits and Sysems in Video Technology*, 6(5):519–521, October 1996.

[7] Villasenor, J., Belzer, B., and Liao, J. "Wavelet Filter Evaluation for Image Compression." *IEEE Transactions on Image Processing*, 2:1053–1060, August 1995.