Fast JPEG Encoding for Color Fax Using HVQ

Ricardo L. de Queiroz¹ and Patrick Fleckenstein²

¹Xerox Corporation, Webster, NY, 14580

`queiroz@wrc.xerox.com`

²Rochester Institute of Technology, Rochester, NY, 14623

`pat@csh.rit.edu`

ABSTRACT

We derive a JPEG compliant image compressor which is based on hierarchical vector quantization (HVQ). The goal is to reduce complexity while increasing compression speed. For each block, the DCT DC coefficient is encoded in the regular way while the residual is mapped through HVQ to a pre-computed bit-stream corresponding to the compressed DCT AC coefficients. Approximation quality is generally good for high compression ratios. Color Fax is the primary application target for the proposed system.

1. INTRODUCTION TO COLOR FAX

Black and white facsimile has been widely used for many years and more recently color facsimile became a standard.¹ Color fax systems transmit full pages at moderate to high resolutions over phone lines. While the transmission speed is typically 14.4 Kbps or 28.8Kbps, one might typically transmit an 8.5×11 inch page at 200 or 300 pixels per inch (ppi). Hence, color image data in excess of 11 MB has to be transmitted over channels of less than 3 KB/sec. Without compression, it would take more than an hour to transmit a single page and it would be impractical to use such a system. In order to keep the transmission time reasonable, images must be compressed at very high ratios.

Color fax relies on a negotiation and transmission protocol which is similar to its black and white counterpart, with a few adaptations for color transmission. The major difference is the image data bitstream. Instead of using the MH1 or MMR compression schemes used for black and white fax, in color fax's baseline mode, the page is compressed using JPEG² in a CIELAB color space. The protocols and transactions between transmitter and receiver make sure that key transmission parameters such as resolution and color space are matched across both systems' capabilities. Summarizing, the low level protocols are very similar to regular fax while the continuous tone image is JPEG compressed instead of using binary compression.

1.1. Timing constraints

As the receiver capabilities are unknown until the negotiation is complete, image processing and compression for transmission is commonly performed on-the-fly. In the negotiation phase the transmitter dials the receiver, the receiver answers and handshaking is initiated. During this phase, there is a well established protocol (just like in the black and white case) to determine what the transmitter's and receiver's capabilities are. Based on the mutually best capability, the image is transmitted using a resolution and bit-depth that does not exceed this capability. For example, a color fax may dial a black and white fax, being, thus, forced to send regular black and white fax data instead of color.

Typically, an image will be scanned, color corrected, scaled if necessary, and compressed for transmission. These steps are computationally expensive and cannot all be done off-line. This is so because the transmission mode is unknown until handshaking is completed and the image parameters (resolution, color space, etc.) may be changed depending on the receiver. The key timing constraint is that about a quarter second after negotiation is completed, the receiver needs to begin receiving data. This is not enough time with today's technology to process the whole image. So, unless one stores the processed data in all possible variations (which is impractical), one has to process and compress the image on-the-fly.

Another very important timing constraint is due to transmission time-outs. If the receiving modem does not receive data for few milliseconds it times out and closes the connection. In order for the modem to transmit data at a constant rate it also has to receive enough data from the JPEG compression module, i.e. buffer underflow can stop the transmission. This is a major timing issue. While JPEG compresses blocks at nearly constant speed it produces a variable number of bits per block. In one experiment, a uniformly distributed white noise image pattern

Figure 1. Bit generation rate for a filtered noisy pattern. The uniformly distributed white noise image pattern is filtered with an averaging filter of given dimension. The image is then JPEG compressed using Quality factor 30, while computing time and resulting file size for each experiment, from which the bit-rate was calculated. (a) Left: normalized compression time; (b) Right: normalized bit rate in bits/sec.

was produced and filtered with an averaging filter of given dimension (from 1 to 100). The image was then JPEG compressed using quality factor 30 (IJG implementation). As the filter size increased, the image became less and less complex and higher and higher compression was achieved. For each test, we computed time and resulting file size, from which the buffer bit-rate was calculated. The Fig. 1(a) shows a plot of the time spent to compress the image as a function of the filter size. Note that in going from white noise (filter size 1) to a very smooth image there is not much difference in computation time. This is because the implementations of operations such as DCT, quantization, etc. are mostly invariant with respect to image contents. However, the number of bits produced by the compression system largely varies from the white noise to the smooth case. The plot of bit rate as a function of filter size is shown in Fig. 1. From this plot one can see that the output bit-rate largely varies as a function of the image complexity.

The worst scenario as far as underflow is concerned for color fax transmission occurs when the image has almost no details, i.e. a flat image. For example, this can occur when closing the scanner's lid without any paper in it, while transmitting the image. It will be a gross error to allow time-outs in such a common scenario. On one hand, one does not want to underflow the buffer to avoid time-outs so that it would be desirable to produce more bits per block in smooth regions. On another hand one wants to produce less bits per block (increase compression) in order to keep the transmission time short. Hence, in order to increase the compressor rate in bits/sec, if one cannot increase the number of bits/block, one has to speed up the compression system. This is the goal of this paper.

1.2. Problem statement

Color fax images are likely to be scanned and printed at high resolution on low cost devices such as multifunction units. The image is to be highly compressed to cope with the channel constraints. It is, thus, desirable to develop a system that

- reduces implementation costs associated with JPEG;
- allows easy implementation in hardware;
- allows fast software implementation in order to cope with the timing constraints;
- approximates a JPEG coder well for high-compression ratios.

We will present a method to approximate JPEG compression which does not involve the traditional JPEG compression steps and is an order of magnitude faster in a software implementation, since it just relies upon look up tables and a few averages for producing the final bitstream. We call this novel technique HVQ-JPEG. Although color fax is a nice application for HVQ-JPEG, its utility is not limited to color fax and can be applied to any other scheme which involves JPEG compression. In fact it can be applied to most transform coders.

Figure 2. General framework for HVQ-based JPEG. DCC is calculated by simple block averaging and encoded in the normal way. The residual is mapped through HVQ into a compressed bit-stream.

2. GENERAL FRAMEWORK

The general idea is depicted in Fig. 2. The image is divided into blocks, as in JPEG.² The DCT DC coefficient (DCC), which is proportional to the average of the block entries, is computed. If the block entries are x_{ij} , and their average is $\bar{x}$, the DCC is found as

$$x_{dc} = \frac{1}{8} \left(\sum_{ij} x_{ij} \right) = 8\bar{x} .$$

The residual data is computed as $y_{ij} = x_{ij} - \bar{x}$ and its transform yields the desired DCT AC coefficients (ACC). In order to generate the compressed data, the ACC and DCC are quantized and encoded. In the case of the DCC, JPEG employs a differential coding approach, so that the quantized DCC is subtracted from its equivalent from a previous block. Thus, DCCs are not independent (see *the JPEG book*² for all details on JPEG). For this reason we quantize and encode the DCC in the same way as JPEG does.

In this paper, the new concept comes from the compression of the ACC. Regular JPEG requires transforming the data, quantization, zigzag scanning, run-length counting, Huffman coding, etc. We avoid all these steps by directly estimating the compressed bit-stream. This is done by applying vector quantization (VQ)³ to map the residual block to an index k , which is then mapped to a compressed bit stream.

In a traditional VQ-based coding system, an input vector (image block) is matched to one of N available representative blocks, for which a codeword or index is generated. Codebooks and representative blocks are known to both transmitter and receiver. The selected codeword is transmitted to a receiver which decodes the codeword and retrieves the associated block as an approximation of the input data block.

Here, as in regular VQ, we match the input block to one entry of a set of N representative blocks B_k . The representative blocks B_k , however, are pre-encoded using JPEG for a particular selection of quantizer and Huffman tables, generating N associated bitstreams S_k of L_k bits each. As the residual block y_{ij} is actually mapped through VQ, only the part of the compressed bitstream corresponding to ACC is retained. The bit streams S_k are pre-stored and represent the ACC compressed data relative to a VQ approximation of the input block, i.e. the residual is mapped directly to S_k . Summarizing, the algorithm is:

Off-line:

- Select a training set of images.
- Remove the mean of each block (i.e. use y_{ij}).
- Design a set of representative blocks $\{B_k\}$ for the training data, for example using the LBG algorithm.³
- Compress each block B_k in the set.

Figure 3. HVQ look-up tables and a 3-level encoding hierarchy.

-
- Discard the compressed DCC and store the bitstream corresponding to the ACC (code S_k of length L_k) in a bit-stream look-up table.

On-line:

- Retrieve input image block.
- Compute scaled average (DCC).
- Quantize and encode DCC as in JPEG.
- Transmit encoded DCC.
- Compute residual (block entries less average).
- Map residual to index k using VQ.
- Retrieve k -th ACC bitstream and transmit it (stream S_k of L_k bits).

3. HVQ BRIEFLY

Although any VQ system could theoretically be employed, hierarchical VQ (HVQ)^{4,5} is the system of choice since it can be implemented solely with 63 look-ups over 6 tables for a block of 8×8 pixels. In HVQ, as illustrated in Fig. 3, two M -bit input symbols are mapped to one B -bit codeword. Next, two codewords are mapped to one codeword in a next level and the process is repeated until there is only one resulting codeword in the block. As pairs of M bit codewords are mapped to a B -bit codeword, a table of 2^M entries of B bits each is generated for each level. An HVQ decoder is also based on look-up tables as well, but it is of no concern to us in this paper as the output of this coder is a JPEG bitstream. In effect, HVQ is only used in this paper as a very fast mapping tool, i.e. to quickly map the input block to an index corresponding to an approximating block.

We refer to our approach, which employs HVQ to map the ACC, as HVQ-JPEG.

Figure 4. (a) Left: peak SNR (PSNR) plots for different bit-rates comparing JPEG (dashed line) and HVQ-JPEG (solid line) for image Lena. (b) Right: same results, after halftoning and inverse halftoning the decompressed images before computing the PSNR.

4. PERFORMANCE EVALUATION

At this stage, we are unable to design useful tables for 6-level HVQ. The reason for this is very common in VQ design. As the vector size increases it is more likely to converge to a point farther away from the global optimal. The extra complexity in this case is because we used a greedy HVQ design, in which codebooks are optimized at each stage. A global (non-greedy) HVQ design has not been reported in the literature so far. For 6 stages the quality deteriorates rapidly. The positive note is that this is a design issue for which there are ongoing studies and we believe the quality of HVQ design for large vectors may improve in the future.

To cope with present design limitations we modify the algorithm to accomodate smaller vectors. We downsample the input block in Fig. 2 (using 2×2 pixel averaging) obtaining the downsampled block samples x'_{ij} whose average is the same of that of x_{ij} . The DCC is then computed as $x_{dc} = \frac{1}{2} \left(\sum_{ij} x'_{ij} \right) = 8\bar{x}$ and the residual is $y_{ij} = x'_{ij} - \bar{x}$. In the design phase, the 4×4 residual patterns are interpolated to reconstruct 8×8 blocks in order to find the library of ACC compressed bit streams S_k . This modification immediately imposes a limitation on the reproduction of high frequency features of the image. However, it does not impact much the overall quality because our target is very high compression wherein high frequency details are not commonly represented.

For simulations, we designed codebooks for HVQ using 10 bit codewords for every stage. Hence, the ACC of the input block are mapped to at most 1024 different patterns. Downsampling and HVQ imprecision (quantization errors) are limiting factors for the HVQ-JPEG performance. However, complexity for HVQ-JPEG is unrivaled. For processing each block, the complete system (which we are not using in these simulations) demands 63 table look-ups (HVQ); 63 additions and one shift (DCC); 64 subtractions (residual); plus the complexity to quantize and encode the DCC. With subsampling, which is the method we are using for simulations, there are only 15 table look-ups (HVQ) and 16 subtractions (residual), while there are 63 additions and 17 shifts (to compute the mean and the subsampled block). As a comparison, a regular JPEG coder would use hundreds of operations only to perform the DCT, plus hundreds of other operations to quantize all 64 coefficients, to zigzag scan the quantized data and to encode the result.

The plot shown in Fig. 4(a) illustrates the objective performance of the HVQ-JPEG against regular JPEG for several bit rates using peak SNR as a distortion measure. The gap in objective performance is significant but the errors can be largely masked for systems like color fax. In these, the images are likely to be scanned and printed at resolutions around 300 pels/inch so that blocks may be smaller than 1 mm ! Also, the image is likely to be halftoned. In order to partially account for the imaging degradation we repeated the objective experiments in Fig. 4(a), this time halftoning and inverse halftoning⁶ the decompressed image before computing the objective distortion. The halftoning and inverse halftoning steps are not critical, serving merely to illustrate that most of the lost details, compared to JPEG, fall within regions of low visibility, e.g. textured patterns, etc. A similar goal could be achieved by filtering both images (original and reconstructed) before computing the PSNR. After processing both images before computing the PSNR, the gap in performance is largely reduced as can be seen in Fig. 4(b). In the example in Fig. 4, we compressed the popular image Lena while the codebook was designed using 7 digitized photos.

Figure 5. From left to right: (a) original image of 400×280 pixels, i.e. image obtained if there was no quantization in JPEG; (b) image obtained if there was no scalar quantization in HVQ-JPEG (PSNR 23.6 dB); (c) JPEG compressed image (PSNR 24.4 dB); (d) HVQ-JPEG compressed image (PSNR 22.4 dB). Compression ratio is 36:1.

Examples are shown in Fig. 5 for a 400-pixel wide, 280-pixel high image which should be printed at a height slightly under 1 inch at 300 ppi. Compression ratio for this simple test is a modest 36:1, which is conservative for color fax applications*. In Fig. 5(a) the original image is shown. Note that this is what one would reconstruct if there was no quantization in JPEG. In Fig. 5(b), the mean-removed HVQ-reconstructed image is shown. This is what one would reconstruct if there is no JPEG quantization (scalar quantization in the JPEG step of the bitstream design phase) in HVQ-JPEG, i.e. it is the best possible reconstruction no matter what quality factor is selected for the JPEG coder. Figure 5(c) shows the reconstructed JPEG image after compression to 36:1. Figure 5(d) shows the reconstructed HVQ-JPEG image after compression to 36:1. Despite the 2dB difference in PSNR between images (c) and (d) in Fig. 5, the image produced by HVQ-JPEG is competitive with its JPEG counterpart. The edges are softer but there are less ringing and blocking artifacts. Overall, the main picture contents are preserved across coders for a casual observer. Similar results are presented in Figs. 6 and 7 for another test image (using the same HVQ tables as in the previous example) and for a more realistic compression of 50:1. Again, the quality of the images compressed with HVQ-JPEG and JPEG are competitive, although HVQ-JPEG can be implemented almost an order of magnitude faster than JPEG.

5. CONCLUSIONS

The proposed system, HVQ-JPEG, is extremely simple to implement as it does not require most of the standard JPEG steps (e.g. DCT, quantization, Huffman coding etc.). The only complex operation is the codebook design which is performed off-line. It is only suitable for high compression because of HVQ's inherent limitations. For large compression ratios, artifacts caused by HVQ mapping start to be masked by the artifacts caused by the scalar quantization in JPEG. As a bonus, since HVQ-JPEG tables are designed by training, one can fine-tune the HVQ-JPEG performance to particular image characteristics (e.g. text) by including characteristic images in the training set. HVQ-JPEG can be applied in systems like color fax and is easily implementable in either hardware or software.

Future research will be focused on codebook design and on the use of multiple codebooks with region adaptation.

REFERENCES

1. ITU-T, *Recommendation T.4*, "Standardization of group 3 facsimile apparatus for document transmission," ANNEX E, *Optional Continuous Tone Colour Mode*, July 1996.
2. W. P. Pennebaker and J. L. Mitchell, *JPEG: Still Image Compression Standard*, Van Nostrand-Reinhold, New York, NY, 1993.
3. A. Gersho and R. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic, Norwell, MA, 1992.
4. P. C. Chang, J. May and R. Gray, "Hierarchical vector quantization with table look-up encoders," *Proc. Intl. Conf. Communications*, Chicago, IL, pp. 1452-1455, 1985.
5. M. Vishwanath and P. Chou, "An efficient algorithm for hierarchical compression of video," *Proc. Intl. Conf. Image Processing*, Austin, TX, Vol. 3, pp. 275-279, 1994.
6. J. Luo, R. L. de Queiroz and Z. Fan, "A robust technique for image descreening based on the wavelet transform," *IEEE Trans. on Signal Processing*, Vol. 46, pp. 1179-1185, Apr. 1998.

*A letter-sized page at 300 ppi yields 8.4 MB per color channel. If channels are compressed at 36:1, with chrominance subsampling one can achieve 72:1 compression and yet it would generate compressed page sizes in excess of 350KB for a color image. At 28.8 Kbps, transmission time will be longer than 2 minutes.

Figure 6. Left: original “kids” image, of 1016×1464 pixels. This is the image that would be obtained if there was no quantization in JPEG. Right: image obtained if there was no scalar quantization in HVQ-JPEG.

Figure 7. Left: reconstructed image “kids” after 50:1 JPEG compression; right: same after 50:1 HVQ-JPEG compression.