Adaptive coding of DCT coefficients by Golomb-Rice codes

N. Memon
Imaging Technology Department
Hewlett Packard Laboratories

Abstract

In this report we apply some of the Golomb-Rice coding techniques that emerged in JPEG-LS, a new standard for lossless image compression, and apply them towards coding DCT coefficients in the lossy JPEG baseline algorithm. We show that this results in significant improvements in performance with limited impact on computational complexity. In fact, one significant reduction in complexity provided by the proposed techniques is the complete elimination of the Huffman tables in JPEG baseline which can be a bottleneck in hardware implementations. We give simulation results, comparing the performance of the proposed technique to JPEG baseline, JPEG baseline with optimal Huffman coding (two pass) and JPEG arithmetic.

1 Introduction

The JPEG committee of the International Standards Organization (ISO) has recently developed a new standard for lossless and near-lossless compression of continuous-tone (2 to 16 bits) still images. A baseline algorithm, called JPEG-LS, has already been completed [1] and is awaiting approval by national bodies. The JPEG-LS baseline algorithm despite being remarkably simple and computationally efficient, provides compression performance that is within a few percent of more sophisticated techniques such as CALIC [10] and UCM [7].

One of the key ideas that emerged in JPEG-LS was a simple and effective technique for Golomb-Rice coding using sequential parameter estimation. This technique was part of the revised HP proposal, LOCO-I_{1p} [9] submitted to the JPEG committee in November 1994. Despite being extremely simple to implement both in software and hardware, the coding performance of the Golomb-Rice coding technique proposed in

[9] proved to be, in practice, within a few percent of some more complex arithmetic coding based techniques and was eventually incorporated into the standard with minor revisions [1].

Before the development of JPEG-LS, the most popular compression standards, like, JPEG, MPEG, H263, CCITT Group 4, have essentially used static Huffman codes in the entropy coding stage. This is because adaptive Huffman coding does not provide enough compression improvement in order justify the additional complexity. Adaptive arithmetic coding, on the other hand, despite being used in standards like JBIG and JPEG arithmetic, has seen little use due to concerns about intellectual property restrictions and also perhaps due to the additional computational resources that are needed, especially in a software implementation. JPEG-LS is the first international compression standard that uses an adaptive entropy coding technique that requires only a single pass through the data and requires computational resources that are comparable to static Huffman coding.

In this paper we apply some of the Golomb-Rice coding techniques that emerged in JPEG-LS towards coding DCT coefficients in the lossy JPEG baseline algorithm. We show that this results in significant improvements in performance with limited impact on computational complexity. In fact, one significant reduction in complexity provided by the proposed techniques is the complete elimination of the Huffman tables in JPEG baseline which can be a bottleneck in hardware implementations. Although we focus our attention on still images and the JPEG baseline algorithm, it should be noted that the techniques proposed in this paper are also potentially applicable to video coding standards like MPEG and H263.

The rest of the paper is organized as follows: in section 2 we show how the DC coefficient within each 8×8 DCT block can be coded more efficiently by simply using JPEG-LS on the sub-image comprising of DC coefficients. In section 3 we develop a context based technique for Golomb-Rice coding of AC coefficients. In section 4 we present a technique for run length coding of zero valued AC coefficients that is very similar to the the runlength coding employed in JPEG-LS. We also show how an adaptive reordering of coefficients prior to run length coding can lead to improved performance. Finally in section 5 we give simulation results, comparing the performance of the proposed technique to JPEG baseline, JPEG baseline with optimal Huffman coding (two pass) and JPEG arithmetic.

Image	JPEG-LS	JPEG-HUFF
balloon	3289	4559
barb2	3807	4930
barb	4131	5282
board	2948	4091
boats	3605	4576
girl	3993	4754
hotel	4172	4933
zelda	3613	5010

Table 1: Bytes needed for coding DC coefficients with JPEG-LS and baseline JPEG Huffman coding technique using default quantization table.

2 Coding the DC coefficient by JPEG-LS

In the baseline JPEG algorithm the quantized DC coefficients are differentially encoded. In other words, the quantized DC coefficient of the previous block is used as the predicted value for the current quantized coefficient. The prediction error is then encoded using a Huffman code. Although the Huffman code can be specified by the user, this requires two passes through the image which may not possible in some applications. Hence often the default static Huffman tables are used resulting in poor performance.

One way to achieve better compression performance while coding the DC coefficients in a single pass is to use JPEG-LS. Specifically, the DC coefficients of an image can be treated as a smaller image that essentially has the same smoothness properties encountered in real images. Hence such a sub-image can be efficiently encoded using JPEG-LS which uses adaptive coding and requires only one pass through the data. Furthermore the additional memory requirements for doing this are limited as JPEG-LS requires less than 2K of memory and only the past $\frac{N}{8}$ DC samples need to be buffered, where N is the number of columns in the image. In Table 1 we show the number of bytes needed to encode the DC coefficients using JPEG-LS and estimated baseline JPEG bytes using the default quantization step size. The test images are the luminance planes of a standard set of 576×720 JPEG test images.

3 Golomb-Rice coding of AC coefficients

In JPEG-LS, prediction errors are encoded using a special case of Golomb codes [2] also known as Rice codes [4]. Golomb codes of parameter m encode a positive integer n by encoding n mod m in binary followed by an encoding of n div m in unary. When $m = 2^k$ the encoding procedure has a very simple realization and has been referred to as Rice coding in the literature, but following [8] we refer to them as Golomb-Rice codes. The key factor behind the effective use of Golomb-Rice codes is the estimation of the coding parameter k to be used for a given sample or block of samples. Rice's algorithm exhaustively [4] tries codes with each parameter on a block of samples and selects the one which results in the shortest code length. This parameter is sent to the decoder as side information. However, in JPEG-LS the coding parameter k is estimated on the fly for each prediction error using techniques proposed by Weinberger et. al, [8]. Despite the simplicity of the coding and estimation procedures, the compression performance achieved is surprisingly close to that obtained by arithmetic coding. In their technique, each prediction error is mapped in an interleaved manner to a positive number and then encoded by using a Golomb-Rice code of parameter $m = 2^k$. The parameter k is estimated by maintaining in each context C, the count N_C of the number of times the context C has been encountered so far and A_C, the accumulated sum of magnitudes of prediction errors within this context C. The coding parameter k is then computed as

$$k = \min\{k' \mid 2^{k'} \cdot N_C \geq A_C\}. \tag{1}$$

The strategy employed is an approximation to optimal parameter selection for this entropy coder. For details the reader is referred to [8].

Essentially the above procedure is using the fact (proven in [8]), that for a discrete Laplacian distribution $\mathcal{L}_{\alpha,\lambda}$ with a probability mass function of the form

$$P(e) = p_0 \cdot \exp(-\lambda)^{|e|}, \quad -\alpha/2 \leq e \leq \alpha/2 - 1$$

a good approximation of the Golomb-Rice parameter k that yields the minimal code length for this distribution is

$$k = \lceil \log_2 (a_{\lambda,\alpha}) \rceil$$

where $a_{\lambda,\alpha}$ is the expected value of the magnitude of a random sample drawn from $\mathcal{L}_{\alpha,\lambda}$.

Now, it has been established empirically by several researchers that the AC coefficients of an 8×8 DCT block are well modeled by a Laplacian distribution (for example, see [11] or [5]). Hence in order to compute the appropriate Golomb-Rice encoding parameter we need a good estimate of the expected magnitude of each AC coefficient. In JPEG-LS, good estimates of the expected magnitudes of prediction errors are obtained by using a large number of contexts. Using large number of contexts for coding AC coefficients within a DCT block can be potentially expensive as we would need a different set of contexts for each of the 63 coefficients. This is because although the distribution of each AC coefficient is Laplacian, the variance of this distribution varies with the coefficient position in the DCT block. Hence we would require separate contexts for each of AC coefficients. Not only would this require an enormous amount of memory, but estimation would be ineffective due to the sparse context problem or high model cost. Hence the challenge is to design a small number of contexts that are able to capture the essential structure within a DCT block and further adapt the estimation procedure within each context.

The distribution of a specific AC coefficient depends on the quantizer step size being used for that coefficient. Hence, clearly the entropy coding of AC coefficients needs to be adapted to the specific quantization table being employed. In addition it is known that DCT blocks containing an edge of the same orientation would have a similar structure. For example, DCT blocks containing a vertical (horizontal) edge have most of their energy in the first row (column) of the block. One way to detect the presence of an edge in the current DCT block is to examine the difference between the DC values of adjacent blocks. Hence, we compute the differences

$$\begin{aligned} d_v &= DC_{i,j} - DC_{i-1,j} \\ d_l &= DC_{i,j} - DC_{i-1,j-1} \\ d_h &= DC_{i,j} - DC_{i,j-1} \\ d_r &= DC_{i,j} - DC_{i-1,j+1} \end{aligned}$$

where i, j are block indices. These differences, multiplied by the DC quantization step size can then be quantized into two levels (low, high), yielding $4^2 = 16$ contexts. In addition the DC value of the current block is quantized into t levels yielding $16t$ contexts. Within each context we, like in JPEG-LS, keep track of the average magnitude of each of the 63 AC coefficients. This can be done by maintaining a count N_C of the number of occurrence of each context C and the accumulated sum

of magnitudes A_C within this context. Doing this requires $63 * 16t + 16t$ words of additional memory. The Golomb-Rice parameter for coding an AC coefficient can then be computed as in JPEG-LS, by the procedure given in (1).

3.1 Scaling the expected magnitude

The memory requirement of the above technique is still prohibitive as the number of contexts is unacceptably large. One way to get good estimates for the Golomb-Rice parameter with a smaller number of contexts is to adaptively compute, a scale factor s for each DCT block based on the sum of the actual magnitudes of the AC coefficients seen so far in the block and the sum of the corresponding expected magnitudes. The expected magnitude of the current AC coefficient can then be scaled by this factor s before computing the Golomb-Rice parameter k for encoding the coefficient. Using this scaling technique along with very few contexts was observed to give results better than those obtained by using significantly larger number of contexts. For example, the best results were obtained by using only d_h and d_v and $t = 2$, that is a total of 8 contexts. The disadvantage of the scaling technique, of course is the division operation needed to compute the scaling factor. It should be noted however, that a majority of AC coefficients would have an expected magnitude of zero, thereby obviating the need for this division operation, as they would be encoded using the runlength coding technique described in the next section.

3.2 Bias cancelation and sign flipping

Although the AC coefficients are modeled well by a zero mean Laplacian distribution, when they are conditioned to a sufficiently large number of contexts, it is seen that the distributions of specific coefficients are not zero mean but reveal systematic biases. In JPEG-LS, these biases are removed by keeping track of the average prediction error in each context and subtracting this from the error prior to coding. We adopt the same technique. Further, small additional improvements are obtained by flipping the sign of the AC coefficient prior to encoding, if the average value is negative. This is a form of sign prediction used in CALIC [10]. Finally, further additional improvements can be obtained by adopting the AC prediction techniques used in MPEG.

4 Coding zero runs

Golomb-Rice codes can be inefficient when coding low entropy distributions because the best coding rate achievable is 1 bit per symbol. Obviously for entropy values of less than 1 bit per symbol, such as would be found in high frequency coefficients, this can be very wasteful and lead to significant deterioration in performance. This redundancy can be eliminated by using *alphabet extension*, wherein blocks of symbols rather than individual symbols are coded, thus spreading the excess coding length over many symbols. This process of clubbing several symbols prior to coding produces less skewed distributions, which is desirable ih this particular situation. In baseline JPEG, alphabet extension is done by the usage of what has been called a 2-D Huffman table that jointly codes an AC coefficient and the number of zero coefficients that follow. In addition a special end-of-block code is used which indicates that the remaining coefficients along the zig-zag scan are all zero.

The techniques used by baseline JPEG are static and do not exploit the fact that the probability of seeing a zero coefficient or a long run of zero coefficients varies in different regions of the image. Smooth regions would contain an abundance of zero coefficeints whereas active regions would not. Hence we use adaptive run length coding to handle the situation more effectively. Specifically, run length mode is triggered when the expected magnitude of an AC coefficient in the current context is less than a certain threshold. The specific run length coding scheme used is the MELCODE described in [6]. MELCODE is a binary coding scheme where target sequences contain a Most Probable Symbol (MPS) and a Least Probable Symbol (LPS). In JPEG-LS, if the current symbol is the same as the previous, an MPS is encoded else a LPS is encoded. Runs of the MPS of length n are encoded using a 1 bit. If the run is of length less than n (including 0) it is encoded by a zero bit followed by the binary value of the run length encoded using $\log n$ bits. The parameter n is of the form 2^k and is adaptively updated while encoding a run. For details of the adaptation procedure and other details pertaining to the run-mode the reader is referred to the draft standard [1].

4.1 Adaptive reordering of coefficients

The problem with the runlength coding described above is that often very short runs are coded since a DCT block scanned in zig-zag order still contains many zero

coefficients interspersed with non-zero coefficients. Runlength coding can be made more effective by scanning all the non-zero coefficients and then the zero coefficients. But the coder does not know in advance where the zero coefficients are located. Of course, the probability of a specific coefficient depends in a global sense on the quantization step size being used for the coefficient. However, the probability of a coefficient being zero also depends on the essential structure of the DCT block which is determined by the presence and the type of edges passing through the block. A good deal of this structure is already captured by the contexts used in the previous section for estimating the magnitude of AC coefficients. Hence, we maintain within each context an ordering that enables us to encode the AC coefficients in increasing (or decreasing) order of magnitude. This is done by maintaining an index array of size 63 for each context. With eight contexts, the additional memory needed is $63*8 = 504$ bytes. In addition after encoding each block, this ordering has to be updated, which in the worst case would require sorting the entire index array. However, since the coding of each block only perturbs the current ordering a little bit, it is observed in practice that restoring the ascending or descending ordering requires only a few passes through the index array.

5 Simulation results

In Tables 2 and 3 we show bit rates achieved with the proposed coding scheme and by an implementation of baseline JPEG (the IJPEG distribution) at various quality factors. It is seen that the proposed scheme appears to give significant improvements over baseline JPEG. Further more the improvement is often more than 30% for low bit rates and drop downs to about 10% for higher bit rates. This is because the static Huffman code being used by JPEG baseline is better suited for moderate to high bit rates. For further comparison we list in Table 4, bit rates for the same test set achieved with the two-pass JPEG baseline that optimizes the Huffman code for the image under consideration. Still we see that the proposed technique consistently performs better over the entire range of bit rates. The difference however, is only about 5% on the average. Finally, in Table 5, we show results obtained with JPEG arithmetic. It is seen that JPEG arithmetic is about 3% better on the average.

	10	30	50	70
baloony	5639	12350	17375	25347
barb2y	14196	31184	42780	59928
barby	13896	31934	43710	59994
boardy	7635	15645	22186	32004
boatsy	10176	22555	31158	43580
girly	10507	23624	32460	44982
goldy	10199	25796	36565	51678
hotely	13094	27477	37332	51950
zelday	7484	16733	23307	33514

Table 2: File size at various quality levels using proposed Golomb-Rice coding techniques for AC coefficients and JPEG-LS for DC coefficient.

	10	30	50	70
baloony	8981	14823	19727	26926
barb2y	17017	34051	46702	64100
barby	16979	34910	46935	62625
boardy	11574	20092	26925	36438
boatsy	13188	25445	34337	46378
girly	12937	25640	34864	47122
goldy	12876	28346	39937	55251
hotely	15965	29989	40257	54311
zelday	10239	18278	25207	35429

Table 3: File sizes obtained by baseline JPEG that uses a fixed static Huffman code.

	10	30	50	70
baloony	6064	12995	18167	25794
barb2y	14265	32727	45750	63188
barby	14316	33144	45828	61668
boardy	8898	17911	24989	34653
boatsy	10700	24014	33451	45777
girly	10322	24134	33939	46631
goldy	10177	26504	38688	54629
hotely	13669	28943	39625	53838
zelday	7414	16511	23861	34289

Table 4: File sizes obtained by two pass baseline JPEG using an optimal Huffman code.

	10	30	50	70
baloony	5411	11908	17016	24416
barb2y	12956	29751	41854	58382
barby	13125	30605	42709	58295
boardy	7713	16406	23348	32704
boatsy	9296	21740	30559	42162
girly	9536	22346	31474	43403
goldy	9127	24209	35359	49841
hotely	12514	26987	37545	51671
zelday	6870	15487	22438	32282

Table 5: File sizes obtained by JPEG arithmetic.

References

[1] ISO/IEC JTC 1/SC 29/WG 1. JPEG LS image coding system. ISO Working Document ISO/IEC JTC1/SC29/WG1 N399 - WD14495,, July 1996.

[2] S. W. Golomb (1966). Run-length codings. *IEEE Transactions on Information Theory*, 12(7):399-401.

[3] S. A. Martucci. Reversible compression of HDTV images using median adaptive prediction and arithmetic coding. In *IEEE International Symposium on Circuits and Systems*, pages 1310–1313. IEEE Press, 1990.

[4] R. F. Rice (1979). Some practical universal noiseless coding techniques. Technical Report 79-22, Jet Propulsion Laboratory, California Institute of Technology, Pasadena.

[5] S. R. Smoot and .L. A. Rowe. Study of DCT coefficient distributions *Human Vision and Electronic Imaging*, Proceedings of the SPIE, vol.2657, pp 403-11, 1996.

[6] S. Ono, S. Kino, M. Yoshida, and T. Kimura. Bi-level image coding with MELCODE - comparison of block type code and arithmetic type code. Proceedings Globecomm 89, 1989.

[7] M. Weinberger, J. Rissanen and R. Arps, "On universal context modeling for lossless compression of gray-scale images", *IEEE Trans.Image Proc.*, 5(4):575–586, 1996.

[8] M. Weinberger, G. Seroussi, and G. Sapiro LOCO-I: A low complexity context-based lossless image compression al gorithm, *Proc. 1996 Data Comp. Conf.*, pp140-149, 1996.

[9] M. J. Weinberger, G. Seroussi, and G. Sapiro. LOCO-I: new developments. ISO Working Document ISO/IEC JTC1/SC29/WG1 N245, November, 1994.

[10] X. Wu and N.D. Memon. Context-based adaptive lossless image coding. *IEEE Trans. Comm.*, 45(4):437–444, April 1997.

[11] G. S. Yovanof and S. Liu. Statistical analysis of the DCT coefficients and their quantization error *Proceedings, 13th Asilomar Conference on Signals, Systems and Computers* pp 601-5 vol.1, 1997