Embedded DCT Image Encoding

Fabrício C. de A. Oliveira and Max H. M. Costa
State University of Campinas (Unicamp), Campinas SP, Brazil

Abstract—**This article introduces a new image coding scheme based on the discrete cosine transform (DCT) applied to blocks of image pixels and on the run-length encoding of the binary digits of the DCT coefficients, bit-plane by bit-plane. The main advantages of this new algorithm are: simplicity of implementation, efficiency superior to JPEG's, in peak-signal-to-noise ratio (PSNR) versus bit rate, fully embedded output sequence, and all the advantages that result from the use of the DCT as the coding transform, as, for example, the possibility to be used as part of a video coder with block motion compensation, such as MPEG.**

Keywords—**DCT, run-length coding, embedded coding, block image coding, JPEG, MPEG.**

I. Introduction

MOST recent achievements in still image compression are based on wavelet transforms (WTs). State-of-the-art algorithms such as EZW [1] and SPIHT [2] render very high distortion-versus-rate performance, as well as subjective quality, with considerably low complexity. However, a major setback of the use of WTs is that the transform is applied to the image as a whole, and not to separate regions of it, as does a block transform such as the discrete cosine transform (DCT). This approach makes it difficult to encode different regions of the image with different techniques, as, for example, when coding a video sequence with a method that takes advantage of the temporal redundancy between frames by predicting some regions or blocks of a frame from neighboring frames. Such a method, as those described in the MPEG (Moving Picture Experts Group) standards 1, 2 and 4 [3][4], will encode regions of the frame that cannot be predicted in a different manner (intracoding) from the way applied to the regions that can be predicted (intercoding).

This article presents an image coding method that uses, as its transform, the DCT, as defined by JPEG's (Joint Photographic Experts Group) standard [5]. Although it reduces the efficiency that could be achieved with the use of a WT, it aggregates the many advantages associated with the DCT. Besides the fact that it is a block transform, it also has the advantage that there are a number of fast algorithms to compute it [6]. Some of these algorithms are already implemented in off-the-shelf digital signal processors.

After DCT-transforming the blocks of the image, the algorithm encodes the resulting coefficients in a bit-plane-by-bit-plane fashion, refining their precision at each turn. This process renders a completely embedded encoded image representation, which means that the rate of the encoding process is determined by simply truncating the encoded sequence in the desired point, when a target rate has been reached, or a desired quality of reconstruction has be achieved.

The encoding of the bits of the DCT coefficients is accomplished using a run-length encoder (RLE), based on Golomb's encoder [7]. In this application, we use an adaptive version of Golomb's RLE encoder.

Section II is focused in the run-length encoding method and introduces its adaptation mechanism. Section III presents the encoding scheme and section IV gives the obtained results. Finally, section V concludes the paper.

II. Adaptive Run-length Encoder

Golomb's run-length encoder (RLE) is the optimum entropy coder for geometrically distributed discrete random variables [7]. A *geometric* random variable $\mathbf{X}$ assumes only non-negative integer values with probabilities

$$P\{\mathbf{X}=\mathbf{k}\} = (1-p)p^k, \quad \text{for k} \geq 0.$$

A sequence $\{\mathbf{X_n}\}$ of geometric i.i.d random variables can be derived from a sequence of Bernoulli i.i.d random variables, with probability of *failure* (or 0) equal to p, by counting the number of *failures* before a *success* (or 1). Thus, the RLE encodes the runs of zeroes that appear between ones in a binary sequence.

The RLE with parameter l encodes a run of k zeros with a binary codeword composed of a sequence of $\lfloor k/l \rfloor$ zeros, where $\lfloor \cdot \rfloor$ denotes the integer part of the argument, followed by 1, which signals the end of the run of zeros, and then, by a codeword representing the rest k modl. This codeword belongs to an optimum code for the random variable $\mathbf{Y}$, defined by

$$\mathbf{Y} = \mathbf{X} \pmod{\mathrm{l}},$$

which assumes values between 0 and $l-1$, with probability distribution

$$P\{\mathbf{Y}=\mathbf{k}\} = \frac{(1-p)p^k}{1-p^l}, \quad 0 \leq k < l.$$

Notice that each zero sent at the beginning represents a run of l zeros.

The parameter l, which is called the *length* of the encoder, is an integer whose value depends on p and is uniquely determined by the inequalities

$$p^l + p^{l+1} \leq 1 < p^l + p^{l-1}. \tag{1}$$

It can be shown that, the optimal encoding for $\mathbf{Y}$, given that p satisfies (1), is to use codewords of length $\lfloor \log_2 l \rfloor$, when

$$\mathbf{Y} < \mathbf{2}^{\lfloor \log_2 \mathbf{l} \rfloor + 1} - \mathbf{l}, \tag{2}$$

and codewords of length $\lfloor \log_2 l \rfloor + 1$, otherwise. Thus, the following scheme is adopted in order to encode $\mathbf{Y}$: first, condition (2) is tested and,

- if true, $\mathbf{Y}$ is encoded as its $\lfloor \log_2 l \rfloor$-bit representation, with the most significant bits sent first,

Fabrício C. de A. Oliveira and Max H. M. Costa are with the Department of Communications (DECOM), School of Electrical and Computer Engineering, Unicamp. Contact: DECOM-FEEC-UNICAMP, C.P. 6101, 13083-970, Campinas SP, Brazil, {fabricio,max}@decom.fee.unicamp.br

- and if not true, the codeword that represents $\mathbf{Y}$ will be the $(\lfloor \log_2 l \rfloor + 1)$-bit integer obtained by

$$\mathbf{Y} + \mathbf{2}^{\lfloor \log_2 \mathbf{l} \rfloor + \mathbf{1}} - \mathbf{l},$$

which is sent to the output starting with the most significant bit.

Notice that no codeword prefix matches another codeword, because the highest $\lfloor \log_2 l \rfloor$-bit integer that can be a codeword in the first case, is $2^{\lfloor \log_2 l \rfloor + 1} - l - 1$, and the lowest $\lfloor \log_2 l \rfloor$-bit prefix of a codeword in the second case is $2^{\lfloor \log_2 l \rfloor + 1} - l$.

For example, for a Golomb's RLE of length 3, $\mathbf{Y}$ can be 0, 1 or 2, which are encoded as 0, 10 and 11, respectively. Table I shows the beginning of the code table for the length-3 RLE.

TABLE I
CODE-TABLE FOR THE RLE WITH LENGTH 3.

Run-length	Codeword
0	10
1	110
2	111
3	010
4	0110
5	0111
6	0010
⋮	

A. Adaptivity

In order to encode a sequence of bits whose probability distribution varies with time, it is necessary to devise a method that adapts the encoder to the varying distribution. The adaptation algorithm used in the adaptive RLE adjusts the length of the encoder based on the behavior of the input sequence. However, it is necessary to guarantee that the information used by the algorithm is available to both encoder and decoder at each position of the encoded sequence.

The length is adjusted in two situations: in the middle of a run of zeros, and after the end of the run. Below, it is described how the adaptation method acts in both cases.

1. When a zero that represents a run of l zeroes is sent/received by the encoder/decoder, the length parameter is incremented by $\lfloor (l+1)/2 \rfloor$.
2. When the codeword representing the rest of the zeroes in the run is sent/received by the encoder/decoder, a new estimate of the average run of zeros $\overline{X}$ is calculated, and the algorithm sets the length of the encoder/decoder to $\lfloor (\overline{X}+1)/2 \rfloor$.

The new estimate of the average run of zeroes is calculated by a 1st-order auto-regressive model described by

$$\overline{X}_{n+1} = \alpha \overline{X}_n + (1-\alpha) X_n,$$

where $\alpha \in [0,1]$ is the memory factor.

III. Coding Scheme

THE DCT transform used in this algorithm is the usual JPEG DCT, as described in [5]. But, differently from JPEG, the DCT coefficients are not quantized after their calculation. The non-uniform quantization applied by JPEG would decrease the algorithm's efficiency, which will be measured in mean-squared-error or peak-signal-to-noise ratio. These criteria are optimized with uniform quantization, which is applied progressively by the algorithm as the coefficients are encoded. However, non-uniform quantization could be used, for example, to improve the algorithm in a HVS (human visual system) sense, by using the JPEG table of quantization coefficients, which prioritizes low frequency coefficients over the high frequency ones, based on the HVS sensitivity to different spatial frequencies.

After applying the DCT, the whole set of coefficients, from all the image blocks that are being coded, are copied to a vector $\mathbf{v}$ in an order determined by the position of each coefficient in its block and the position of its block in the image (see III-B).

Then, the vector of ordered coefficients is split into several binary vectors $\mathbf{v_n}$, with $n = 0, 1, \ldots, N$, where each of which contains the n-th bits of the binary representation of the coefficients in $\mathbf{v}$ (see Fig. 2). The zeroth vector $\mathbf{v_0}$ is taken to be the binary vector corresponding to the most significant bit of the highest amplitude coefficient, and the next vectors correspond to the following less significant bits, including bits after the binary point. The position of the zeroth bit is sent to the output in the beginning of the coding process.

Finally, the binary vectors $\mathbf{v_n}$ are encoded one by one, as described in sub-section III-C, using the adaptive run-length encoder described in section II, until a target bit-rate is reached or until the DCT coefficients can be reconstructed at the decoder with a predetermined precision.

Further details of the coding scheme are described in the following sub-sections.

A. Representation of DCT coefficients

The transform coefficients will be represented in a sign-plus-amplitude fixed-point binary form, which is necessary for the subsequent encoding operation. This representation may be built into the DCT transformer using fixed-point arithmetic, which renders much faster encoder and decoder implementations, as compared to those using floating-point arithmetic.

The number of bits necessary to represent the DCT coefficients in fixed-point arithmetic depends on the number f of bits preserved to the right of the binary-point and on the maximum possible absolute value of the coefficients. Considering that the total energy of an 8×8 block of image pixels is limited, and it is given by the sum of the squares of its 64 DCT coefficients, i.e.,

$$E = \sum_{i=0}^{63} |c_i|^2 .$$

A DCT coefficient c_k will have maximum amplitude, for a given value of energy, if all the block's energy is concentrated on that coefficient. In this case,

$$c_k = \pm\sqrt{E}.$$

Therefore, the maximum amplitude of a DCT coefficient is equal to the square-root of the maximum energy, i.e.,

$$|c_i| \leq \sqrt{E_{\max}}.$$

In this article, the values of pixels are represented by 8 bit integers ranging from -128 to 127. The energy of a block is maximum when all of its pixels have maximum amplitude, i.e., 128. Hence,

$$\begin{aligned} E_{\max} &= 64 \cdot 128^2 \\ &= 2^{20}. \end{aligned}$$

Therefore, the maximum amplitude of a DCT coefficient is 2^{10}, and the number of bits necessary to represent it in fixed-point is $10 + f$ bits for the amplitude, plus 1 bit for the sign.

Tests have shown that 3 bits to the right of the binary point ($f = 3$) is enough precision to obtain an almost perfect reconstruction of the image from the fixed-point representation of its DCT coefficients. Actually, with this precision, the reconstruction was perfect for all images tested by the authors. Hence, 14 bits are enough to represent any DCT coefficient with this precision.

The proposed coding method sends to the output the position of the most significant (non-null) bit of the coefficient with the highest absolute value. By the discussion above, it can be seen that 4 bits are necessary to transmit that position.

B. Ordering the Coefficients

The energy compaction produced by the DCT transform concentrates the energy of the blocks in the low-frequency coefficients. For natural images, the variance of the coefficients tend to decrease as frequency increases [8]. However, images also contain object details, such as borders, which do not have the same frequency distribution as most blocks. These details usually produce coefficients with energy closer to equally distributed or, at least, that do not decrease fast as frequency increases. Thus, the magnitude of high-frequency coefficients usually tend to have higher correlation with the magnitude of neighboring high-frequency coefficients, i.e., when a high-frequency coefficient has high amplitude, we can expect that its neighbors will also have high amplitude. The probability of having one isolated high frequency coefficient with high amplitude is, as far as our assumptions go, small.

Thus, the ordering of the coefficients aims not only at organizing the coefficients in an order of decreasing variance, based on their frequency, but also tries to keep coefficients with high correlation of magnitude, together. That is why, as described below, the ordering process keeps high frequency coefficients of the same block together.

B.1 Ordering Algorithm

Let B_k, $i = 1, 2, \ldots, K$, denote the K blocks of a still image, or the blocks of a video frame which will be intracoded by the method. The order of the blocks should keep neighboring blocks together. Fig. 1 shows how the DCT coefficients are indexed in a block. Each coefficient has a two-tuple index $m.n$, where each value of m represents a sub-block of coefficients, in a total of ten (see Fig. 1). The ordering algorithm is shown below.

1. Create an empty vector $\mathbf{v}$ to receive the DCT coefficients.
2. For each sub-block m, from 1 to 10 do
 (a) for each block B_i do
 i. for each coefficient n in sub-block m, add it $(m.n)$ to the end of $\mathbf{v}$.

1.1	2.1	5.1	5.2	8.1	8.2	8.5	8.6
3.1	4.1	5.3	5.4	8.3	8.4	8.7	8.8
6.1	6.2	7.1	7.2	8.9	8.10	8.13	8.14
6.3	6.4	7.3	7.4	8.11	8.12	8.15	8.16
9.1	9.2	9.5	9.6	10.1	10.2	10.5	10.6
9.3	9.4	9.7	9.8	10.3	10.4	10.7	10.8
9.9	9.10	9.13	9.14	10.9	10.10	10.13	10.14
9.11	9.12	9.15	9.16	10.11	10.12	10.15	10.16

Fig. 1. Ordering of the DCT coefficients

C. Encoding Algorithm

After the ordering process, the vector of DCT coefficients $\mathbf{v}$ is split into the binary vectors $\mathbf{v_n}$, $n = 0,\ 1, \ldots, N$, with $\mathbf{v_n}$ containing the n-th bits of the DCT coefficients, as shown in Fig. 2. These vectors are then encoded according to the following algorithm:

1. Create vector $\mathbf{s}$ with the same length of $\mathbf{v}$ to save the state of significance of the coefficients during the process;
2. set all elements of $\mathbf{s}$ to *insignificant*;
3. for $n = 0,\ 1, \ldots$, and while the target bit-rate or precision is not reached, do
 (a) initialize the RLE to length 1;
 (b) split $\mathbf{v_n}$ into two binary vectors, $\mathbf{u_n}$ and $\mathbf{w_n}$, with $\mathbf{u_n}$ containing the bits of the coefficients that are not significant yet, according to $\mathbf{s}$, and $\mathbf{w_n}$ containing the other bits of $\mathbf{v_n}$.
 (c) encode $\mathbf{u_n}$ using the RLE;
 (d) send the signs of the coefficients which are *insignificant* and whose bit in $\mathbf{u_n}$ is not zero;
 (e) send $\mathbf{w_n}$ to the output "as is", without encoding;
 (f) set to *significant* the elements of $\mathbf{s}$ which correspond to the bits of $\mathbf{v_n}$ that are not zero;

Vector $\mathbf{s}$ saves the state of *significance* of the coefficients during the process. In the beginning, the states of all coefficients are set to insignificant. A coefficient becomes significant after its most significant bit is encoded.

The algorithm stops when a target bit-rate or precision is achieved.

IV. Results

Fig. 3 shows the PSNR (peak-signal-to-noise ratio) versus bit-rate results of the algorithm as compared to results obtained with JPEG when applied to the Lena image. JPEG's algorithm is applied using uniform quantization in order to maximize the efficiency according to the PSNR criterion.

PSNR measures the ratio between maximum signal amplitude and root-mean-squared error (RMSE), and is given, in dB, by:

$$PSNR = 20 \log_{10} \left(\frac{255}{RMSE} \right),$$

where the RMSE is the measured root-mean-squared error between the original image and its reconstructed representation,

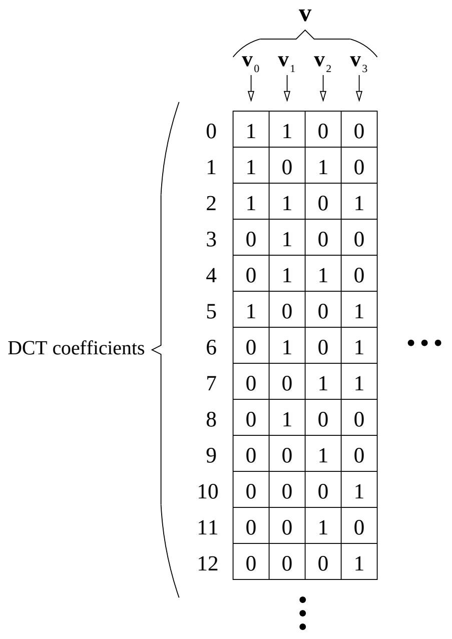

Fig. 2. Splitting of $\mathbf{v}$ into binary vectors $\mathbf{v_n}$.

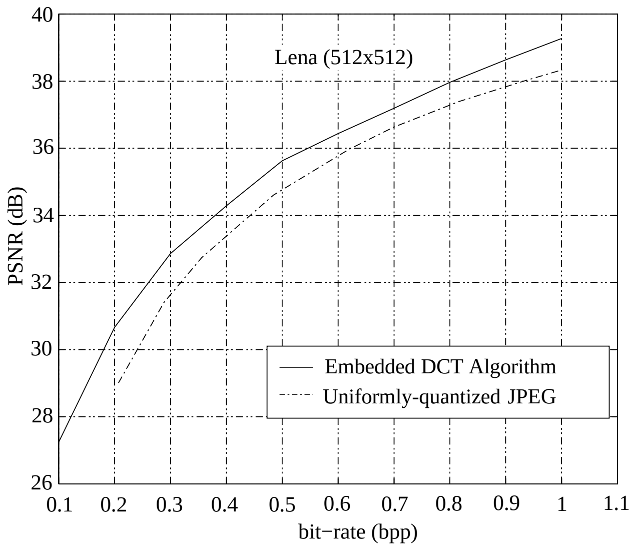

Fig. 3. PSNR-versus-bit-rate curve achieved by the algorithm for image Lena compared to JPEG's with uniform quantization.

i.e.,

$$RMSE = \frac{1}{M \cdot N} \sum_{i,j=0}^{M,N} [\hat{\mathbf{X}}(i,j) - \mathbf{X}(i,j)]^2.$$

The proposed image encoding algorithm shows good performance in terms of PSNR versus bit-rate. A gain of about 1 dB over JPEG is found for the Lena image.

V. Conclusions

A DCT-based image compression technique has been presented. The algorithm uses an adaptive version of Golomb's run-length encoder. The rate $\times$ distortion performance observed for the Lena image, shows a gain of about 1 dB with respect to JPEG with uniform quatization. One of the advantages of DCT-based compression schemes is that they may be well adapted to work video frames, using motion estimation and compensation.

Acknowledgments

The authors would like to acknowledge the Independent JPEG Group (IJG) for its software implementation of the standard JPEG encoder and decoder that is freely distributed. More information about IJG can be found at *http://www.ijg.org*.

References

[1] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 41, no. 12, pp. 3445–3462, 1993.
[2] A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 243–250, 1996.
[3] MPEG-1 Standard Draft, "MPEG-1: Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s", *International Organization for Standardization ISO/IEC JTC1/SC29/WG11N*, 1992.
[4] "Generic Coding of Moving Pictures and Associated Audio", *ISO/IEC 13818 (MPEG-2 standard)*, 1994.
[5] G. K. Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, vol. 34, pp. 30–44, April 1991.
[6] C. Loeffler, A. Ligtenberg, and G. Moschytz, "Practical fast 1D DCT algorithms with 11 multiplications," 1989.
[7] R. G. Gallager and D. C. van Voorhis, "Optimal source codes for geometrically distributed integer alphabets," *IEEE Transactions on Information Theory*, vol. IT-21, pp. 228–230.
[8] N. S. Jayant and P. Noll. *Digital Coding of Waveforms*. Englewood, NJ: Prentice-Hall, 1984.