DCT COEFFICIENT PREDICTION FOR JPEG IMAGE CODING

Gopal Lakhani

Texas Tech University, Lubbock Texas, 79409-3104, USA
(lakhani@cs.ttu.edu)

ABSTRACT

The JPEG baseline algorithm follows a block-based coding approach and therefore, it does not explore source redundancy at the sub-block level. This note explores correlation between adjacent rows (or columns) at the block boundaries for predicting DCT coefficients of the first row/column of DCT blocks. Experimental results show that our prediction method reduces the average JPEG DC residual by about 75% for images compressed at the default quality level. The same for AC01/10 coefficients is about 30%. It reduces the final code bits by about 4.55% of the total image code for grey images. Our method can be implemented as a part of the JPEG codec without requiring any changes to its control structure or to its code stream syntax.

Keywords - Image coding, DCT, prediction methods, JPEG

1. INTRODUCTION

The JPEG baseline algorithm, [1], follows a block-based compression approach. It divides the input image into 8x8 pixel blocks, transforms each block using DCT, and then codes the DC and AC coefficients. To reduce redundancy due to correlation between pixels of adjacent blocks, it predicts only the DC coefficient. For intra-block redundancy, it only quantizes the DCT coefficients and does not use predictive coding. In other words, it explores spatial redundancy at the block level and not at the pixel level. Consequently, the JPEG algorithm leaves a fair amount of spatial redundancy unexplored. In this note, we consider correlation between adjacent rows and/or columns at block boundaries and develop a new method for DCT coefficient prediction. The problem is that DCT coefficients of different frequencies are mutually independent, but the prediction residual of a coefficient must be expressed in terms of coefficients of other frequencies in order to compute the residuals in the transform domain. We overcome this difficulty. The main attraction of our method is that it can be implemented without requiring any changes to the JPEG codec structure or to the JPEG code bit stream syntax.

DCT restoration has been studied widely in context of variety of still image and video compression problems such as blocking artifact reduction, image resizing, and transmission error recovery. However, most of these studies can not be used for DCT prediction due to the following constraints. (1) A JPEG codec must compute prediction residuals in the raster-scan order of blocks and therefore, only two sides of a block are accessible for DCT prediction (blocking reduction methods consider all four sides). (2) A JPEG codec must use only the DCT coefficients and not pixels of a block for prediction. The reason is that the JPEG standard does not require any specific implementation of discrete cosine transform in order to avoid different DCT implementations reconstructing pixels of subsequent blocks differently. (3) A JPEG prediction algorithm must use the same computation for all blocks, because the JPEG standard has no provision for coding any control information for individual blocks. Therefore, we review literature on the DCT *prediction* problem only.

A prediction method for block-based coding scheme must explore spatial redundancy at one of the following three levels: pixel level, row/column level, or block level. JPEG reduces spatial redundancy at the block level and uses only DC coefficient prediction for this purpose. MPEG-4 also explores at the block level, but it predicts both DC and AC coefficients. [2] proposes the following adaptive prediction method for MPEG-4. Let C, B, A and X denote blocks of a 2x2 block array shown in Fig. 1, where C, B and A are already coded and X is to be coded next.

$$\begin{aligned} if \quad & (|DC(A) - DC(C)| < |DC(C) - DC(B)|) \\ & \qquad pred(DC(X)) = DC(B) \\ else \quad & \qquad pred(DC(X)) = DC(A) \end{aligned} \tag{1}$$

If A is the source for DC prediction, it tests the following AC prediction criterion:

$$\left(\sum_{i=1}^{7} |AC_{i,0}^{X}| - \sum_{i=1}^{7} |AC_{i,0}^{X} - AC_{i,0}^{A}| \right) > 0 \tag{2}$$

If (2) holds, a AC_{pred} flag is set in the macro-block header and only the first row AC coefficients of X are predicted from the co-located first row AC coefficients of B. H.263+ also uses this method. We can not use this approach for JPEG algorithm, because there is no provision of any flags in the JPEG code syntax for individual blocks.

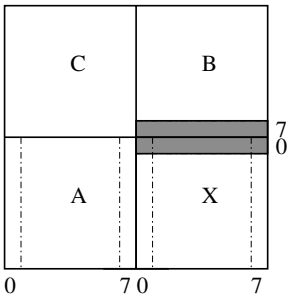

Fig. 1. Exploring inter row/column redundancy for DCT

H.264/AVC is a recent standard for video coding [3]. It applies predictive coding in the pixel domain. This is possible, because its block transformation is reversible and therefore, H.264 decoder can recreate the original pixels of the image without any losses from the given transform coefficients. Another major difference is that H.264 computes correlation between boundary pixels in nine different ways and codes the best choice as an extra information for each block. Yet another major difference is that H.264 applies prediction mostly for 4x4 pixel blocks to get maximum advantages. Obviously, none of these ideas can be explored in the JPEG algorithm due to restrictions noted above as (1)-(3).

An alternative is to explore correlation between the DCT coefficients of the same frequency only, i.e., construct subband images each comprising of a single frequency coefficient (e.g., AC_{01}) and explore predictive coding for subband images. [4] uses this approach for exploring inter-block redundancy, but they use bit-plane entropy coding algorithm, not followed in the JPEG algorithm. [5] gives a constructive method for estimating AC coefficients in the DCT domain.

In this note, we explore spatial redundancy at the row/column level. We first predict the AC coefficients and then use them to predict the DC coefficient. The core of our prediction method is to explore similarities between adjacent rows or columns at the boundary of X with A and B in terms of 1-D DCT coefficients of these rows or columns and then express them in terms of 2-D DCT coefficients of X, A, or B. Section II presents our prediction equations, and Section III presents experimental results on reduction in the average coefficient size and in the final code size. Experimental results show that our method obtains huge reductions; for example, we reduce the average JPEG DC residual by 75% and AC_{01} (AC_{10}) residuals by 30%.

2. DCT PREDICTION

We compressed eleven widely used test images at the default quantization level. A montage of these images except Lena is given in Fig. 2. The average size of DCT coefficients for all positions $(i, j) : 0 \leq i, j \leq 7$, computed by $100 * |DCT_{i,j}|/(\sum_{p,q=0,0}^{7,7} |DCT_{p,q}|)$, is given in Table I;

Table I - Average DCT coefficient size

	0	1	2	3	4	5	6	7
0	16.2	15.5	8.4	3.2	1.4	0.5	0.2	0.1
1	13.7	7.4	4.0	1.9	0.9	0.2	0.1	0.0
2	5.3	3.8	2.7	1.2	0.5	0.2	0.0	0.0
3	3.2	1.8	1.2	0.6	0.2	0.1	0.0	0.0
4	1.6	1.0	0.4	0.2	0.0	0.0	0.0	0.0
5	0.8	0.4	0.1	0.1	0.0	0.0	0.0	0.0
6	0.2	0.1	0.0	0.0	0.0	0.0	0.0	0.0
7	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0

$DCT_{0,0}$ is differentially coded. The table shows that about 60% of the block energy is stored in just five coefficients, DC, $AC_{01}, AC_{10}, AC_{02}$, and AC_{20}. Also, any image continuity in the horizontal and/or vertical direction between adjoining blocks is absorbed in the first row/column DCT coefficients only. Hence, we concentrate on the prediction of these five coefficients only.

2.1. Computing 1-D DCT from 2-D DCT

Let $C_{u,x} = c(u)\, cos((2x + 1)u\pi/16)$ denote the basis elements of the discrete cosine transform, where $0 \leq x,\ u \leq 7$, $c(0) = 1/(2\sqrt{2})$, and $c(u) = 1/2$ for $u \neq 0$. Let $\{f_{x,y} : 0 \leq x, y \leq 7\}$ denote the pixels of an image block f and $\{F_{u,v} : 0 \leq u, v \leq 7\}$ denote its DCT block. Then

$$F_{u,v} = \sum_{y=0}^{7} (C_{v,y} \sum_{x=0}^{7} f_{x,y} C_{u,x}). \tag{3}$$

The DCT is an unitary orthogonal transform. Consequently,

$$\sum_{x=0}^{7} C_{u,x} C_{v,x} = \begin{cases} 0 & u \neq v \\ 1 & u = v. \end{cases} \tag{4}$$

Rewriting (3) using (4),

$$\sum_{y=0}^{7} F_{u,y} C_{v,y} = \sum_{x=0}^{7} f_{x,v} C_{u,x} \tag{5}$$

We interpret (5) by stating that the u^{th} 1-D DCT coefficient of a column (row) of an image block can be computed from the u^{th} row (column) of the 2-D DCT of that block.

2.2. Our DCT prediction equations

Let $x_{i,j}$ denote pixels and $X_{u,v}$ denote the 2-D DCT coefficients of the block X, where $0 \leq i, j, u, v \leq 7$. Let $X_u^{(v,-)}$ and $X_u^{(-,v)}$ denote the u^{th} 1-D DCT coefficient of the v^{th} row and column of X, respectively. Let $A_{u,v}, A_u^{(v,-)}, A_u^{(-,v)}$, $B_{u,v}, B_u^{(v,-)}$, and $B_u^{(-,v)}$ be defined likewise for blocks A and B, respectively. Adjacent pixels in a typical photographic image are mostly similar. Therefore, there should

Fig. 2. Our test images

exist significant similarities between column 7 of A and column 0 of X and between row 7 of B and row 0 of X (see Fig. 1). Consequently, $A_u^{(-,7)}$ and $X_u^{(-,0)}$ are likely to be close, especially for small u. Using (5) and that $cos(15u\pi/16) = (-1)^u cos(u\pi/16), 0 \leq u \leq 7$, the difference between the two coefficients is

$$\begin{aligned}(X_u^{(-,0)} - A_u^{(-,7)}) &= \sum_{y=0}^{7}(X_{u,y}C_{0,y} - A_{u,y}C_{7,y}) \\ &= \sum_{y=0}^{7}(X_{u,y} - (-1)^y A_{u,y})C_{0,y}\end{aligned}$$

$$\begin{aligned}2\sqrt{2}(X_u^{(-,0)} - A_u^{(-,7)}) = (X_{u,0} - A_{u,0}) + \\ \sqrt{2}\sum_{y=1}^{7}(X_{u,y} - (-1)^y A_{u,y})cos((2y+1)u\pi/16)\end{aligned} \quad (6)$$

Since the JPEG entropy encoder codes only integer data and that the encoder and the decoder must compute the same prediction, (6) should be expressed for integer domain arithmetic. We multiply the $cosine$ terms by a large integer 2^{13} and truncate them to the nearest integers.

$$pD = (X_{u,0} - A_{u,0}) + (\sum_{y=1}^{7}(X_{u,y} - (-1)^y A_{u,y})I_y)/2^{13} \quad (7)$$

where $I = \{11363, 10703, 9633, 8192, 6436, 4433, 2260\}$. Note that this simplification of (6) has no impact on the decoded value of $X_{u,0}$, because both the encoder and decoder apply the same formulation (7). The predictive residual, pD, is generally close to $2\sqrt{2}(X_u^{(-,0)} - A_u^{(-,7)})$ (our codec does not compute this 1-D DCT difference; it is given to interpret pD.).

The first row coefficient $X_{0,u}$ is predicted similarly using B in place of A and interchanging row and column prefixes in (7).

It is easy to verify that use of (7) causes no losses, i.e., our predictive coding is lossless. Moreover, it computes prediction in the transform domain. It essentially captures the low-frequency difference left out by taking $(X_{u,0} - A_{u,0})$ from the difference of the column 0 pixels of X from column 7 pixels of A and represents it in terms of higher frequency AC coefficients. This is the main contribution of this paper. (6) is formulated to explore inter-block redundancy. For intra-block redundancy reduction, it can be reformulated to compute differences between columns (or rows) of the same block.

Our codec works as follows. The encoder codes the interior AC coefficients $X_{u,v}$, $u > 0, v > 0$, directly (no prediction). It codes pD in place of $X_{u,0}, u > 0$ for the first column AC coefficients. For $X_{0,0}$, there is a choice because it can also be predicted using the first row AC coefficients $X_{0,v}, v > 0$. Our encoder codes the average of the two predictions to take the advantage of pixel correlation in both directions. The decoder first restores $X_{u,0}$ and $X_{0,u}$ for $u > 0$, because it can compute the second term of the right side of (7); note that $X_{u,y}$ and $A_{u,y}$ are known to the decoder for $u > 0, y > 0$. The decoder restores $X_{0,0}$, at the end.

3. EXPERIMENTAL RESULTS

We used images of Fig. 2 for performance evaluation of our prediction method. Name and size of these images are given in Columns 1-2 of Table II in the same order as shown in Fig. 2. Three sets of experimental results are given to measure advantages of our method over the JPEG algorithm. Results for AC01 and AC10 are given together (same with AC02 and AC20), because their semantic is same. (7) was applied to quantized DCT coefficients. Results were computed only for quality level 75. The reason is that JPEG multiplies defaults quantizers by a scalar factor to compute quantizers for other quality levels and since (7) is linear, the prediction residuals would reduce by the same factor and

Table II - Comparing with JPEG DCT coding

Image	Size	DCT Coeff. ratio			Code size ratio		% code
		DC	AC01/10	AC02/20	DC	AC01/10	saving
1	2	3	4	5	6	7	8
Peppers	512x512	0.2085	0.5152	0.8300	0.6691	0.8494	5.49
Mandrill	512x512	0.6429	1.2775	1.6723	0.8903	1.0641	0.60
Tiffany	512x512	0.2838	0.6799	0.9001	0.6851	0.9020	4.31
Elaine	512x512	0.1991	0.6120	0.9200	0.6542	0.8955	4.68
Goldhill	720x576	0.4225	0.6620	0.8805	0.8060	0.9238	3.25
Girl	720x576	0.2743	0.6840	0.9005	0.7095	0.9109	4.53
Zelda	720x576	0.1923	0.4750	0.8500	0.6181	0.8184	10.16
Barbara	720x576	0.3346	0.9023	1.2301	0.7608	0.9578	2.60
Monarch	768x512	0.2323	0.6140	0.8342	0.6602	0.8943	5.47
Tulips	768x512	0.2290	0.6070	0.6401	0.6976	0.9552	4.31
Lena	512x512	0.2028	0.6585	0.8701	0.6617	0.8957	4.76
Average		0.2929	0.6988	0.9038	0.7120	0.9152	4.55

we would obtain the same reduction ratios.

Columns 3-5 of Table II give the first set of results. Column 3 shows the ratio of our DC residuals with the JPEG DC residuals. The average of this column is 0.2929, which means that (7) reduces the average JPEG DC residual by about 71%; it is a phenomenal reduction. Columns 4 and 5 gives ratio of predicted coefficients with the original AC coefficients. (7) reduces the AC01 and AC10 coefficients by about 30% on the average. Note that AC coefficients depend more on the contents of the block and much less on the surrounding block pixels in comparison to the DC coefficient. It is the reason that our prediction residuals of AC01/10 are larger than the original coefficients for Mandrill, because there is practically no correlation between adjacent pixels. Still a 30% reduction is excellent and prediction coding of AC01 and AC10 is desired because together they constitute about 29% of the total image energy (see Table I).

Columns 6-7 give the second set of results and report the ratio of the code size of the DC and AC01/10 with the JPEG code size of same frequency, respectively. We used the JPEG arithmetic coding [6] because it is more suited for code size comparison than the Huffman coding due to the following two reasons: (1) since distribution of residuals is different from the original DCT coefficients, we should not use the same Huffman code tables, and (2) the arithmetic coding is adaptive. Before reporting savings of our method, we first present an estimation of the reduction in the entropy. Our experiments show that the average code size per JPEG DC residual is 6.35 bits, including the sign bit. It means that a 70% reduction to the average JPEG DC residual should reduce its code size by 1.8 bits and we should expect a reduction of about $1.8/6.35 \approx 28.3\%$ in the JPEG DC code size. The average of Column 6 is 0.7120, which means that our method generates about 28.8% smaller DC code than JPEG; it matches with our estimation. Column 7 shows that average reduction in the code size of AC01/10 coefficients in about 8.5%, which is still significant. The reduction in the AC02/20 code size is quite small (just 1.2%) and hence it is not reported for individual images.

To summarize, this note shows that at the expense of a few additional arithmetic operations, we can predict significant DCT coefficients easily to reduce the image code size. Our last set of results (given in column 8 of Table II) shows that on the average, we can save about 4.55% of the total image code size. This saving should be judged considering that our method does not incur any losses to DCT coefficients, we require no changes in the JPEG structure and that lossless predictive coding gains are typically very small. Note that we can not apply any pixel-level prediction approach like H.264 for JPEG coding. Hence, the row/column based prediction is the next best option, developed in this note.

4. REFERENCES

[1] W. B. Pennebaker, & J. L. Mitchell, *JPEG Still Image Data Compression* Van Nostrand Reinhard, New York (1993).

[2] A. Puri, R. L. Schmidt, & B. G. Haskell, "Improvements in DCT based video coding," *Proc. SPIE*, vol. 3024, pp. 676-688, Feb. 1997.

[3] G. J. Sullivan & T. Wieland, "Video Compression - From concept to the H.264/AVC standard", *Proc. IEEE*, vol. 93, pp. 18-35, Jan. 2005.

[4] C. Tu, & T. D. Tran, "Context-based entropy coding of block transform coefficients for image compression," *IEEE Trans. Image Proc.,* Vol. 11, pp. 1271-84, Nov. 2002.

[5] G. A. Triantafyllidis, D. Tzovaras, & M. G. Strintzis, "Blocking artifact detection and reduction in compressed data", *IEEE Trans. Circuits Sys. Video Tech.,* vol. 10, pp. 877-890, Oct. 2002.

[6] G. Vollbeding, "JPEG Arithmetic coding Software," 1998 (http://sylvana.net/jpeg-ari/).