QUANTIZED DISCRETE COSINE TRANSFORM: A COMBINATION OF DCT AND SCALAR QUANTIZATION

Khanh Nguyen-Phi, Alen Docef and Faouzi Kossentini

Signal Processing and Multimedia Group,
University of British Columbia
2356 Main Mall, Vancouver BC, V6T1Z4 Canada
knguyen@ece.ubc.ca

ABSTRACT

A typical MPEG-2 video encoder requires that DCT and quantization be performed in most cases. In this paper, we show how to combine these two steps, reducing sunstantially the number of computations. The new nonlinear transform is called the Quantized Discrete Cosine Transform, or QDCT. We also introduce a new method to trade-off the computational complexity and the precision of the QDCT. Although the QDCT is independent of input data, better trade-offs can be obtained by making it data dependent, which is appropriate in multimedia applications such as MPEG-2 video coding. The results presented in this paper can also be extended to other linear transforms and/or other coding methods.

1. INTRODUCTION

Most popular techniques for image and video compression are based on the same hybrid motion compensated prediction and DCT video coding method. After the first step of motion compensation, error (difference) data of image or video are decorrelated by the Discrete Cosine Transform (DCT)[3]. The transformed coefficients are then quantized by a scalar or vector quantizer, although simpler scalar quantization is almost always used. Finally, the quantized coefficients are coded by an entropy coder.

In this paper, we show how to combine the DCT and the quantization steps into one, thus saving many computations. The new quantized DCT (QDCT) is targeted for MPEG-2 video encoding, but the underlying method can also be extended to other linear transforms and other compression methods as well. The paper is structured as follows. In Section 2, we derive the simple equation of the QDCT. In Section 3, we discuss the implementation of the QDCT using integer arithmetic and its direct application in MPEG-2 for encoding motion compensated prediction difference blocks and present some corresponding experimental results. In Section 4, the computational complexity of QDCT is evaluated within various implementation frameworks. In Section 5, we introduce a new method to trade-off the number of computations for quality, espressed here in term of PSNR. The last section presents some conclusions.

2. QUANTIZED DCT (QDCT)

2.1. One-dimensional QDCT (1D-QDCT)

The 8-point 1D-DCT of $(x_0, x_1, \ldots, x_7)$ is defined by [2]

$$y_k = \frac{c_k}{2} \sum_{i=0}^{7} x_i \cos \frac{(2i+1)k\pi}{16} \qquad (1)$$

where k=0,...,7 and

$$c_k = \begin{cases} \frac{1}{\sqrt{2}} & : \ k = 0 \\ 1 & : \ k \neq 0. \end{cases}$$

Thus, if the scalar quantization process is espressed as

$$y_k^q = \lfloor \frac{y_k}{q} \rfloor, \qquad (2)$$

where $\lfloor \rfloor$ denotes the rounding operators, the quantized 1D-DCT (1D-QDCT) can be attained using

$$y_k = \lfloor \sum_{i=0}^{7} c^q_{ki} x_i \rfloor, \qquad (3)$$

where

$$c^q_{ki} = \frac{c_k \cos \frac{(2i+1)k\pi}{16}}{q}.$$

The corresponding matrix representation yields

$$\mathbf{Y} = \mathbf{CX} \quad \text{and} \qquad (4)$$

$$\mathbf{Y^q} = \lfloor \mathbf{C^q X} \rfloor, \qquad (5)$$

where $\mathbf{Y}$, $\mathbf{Y^q}$ and $\mathbf{X}$ are 8×1 column vectors, $\mathbf{C}$ and $\mathbf{C^q}$ are 8×8 transform matrixes. $\mathbf{C^q}$ can be derived from $\mathbf{C}$ for each value of q, and $\mathbf{C}$ can be fully described by 7 parameters (a,b,c,d,e,f,g) as follows,

$$\mathbf{C} = \begin{bmatrix} g & g & g & g & g & g & g & g \\ a & b & c & d & -d & -c & -b & -a \\ e & f & -f & -e & -e & -f & f & e \\ b & -d & -a & -c & c & a & d & -b \\ g & -g & -g & g & g & -g & -g & g \\ c & -a & d & b & -b & -d & a & -c \\ f & -e & e & -f & -f & e & -e & f \\ d & -c & b & -a & a & -b & c & -d \end{bmatrix}.$$

This means $\mathbf{C^q}$ is also fully described by the corresponding 7 parameters: $a_q = a/q, b_q = b/q, \ldots, g_q = g/q$.

2.2. Two-dimensional QDCT (2D-QDCT)

In a similar manner, we can also write the necessary equations in the 2D case, using the separability of the underlying DCT. They are

$$\mathbf{Y} = \mathbf{C_r X C_c^T} \quad \text{and} \tag{6}$$

$$\mathbf{Y^q} = \lfloor \mathbf{C_r^q X (C_c^q)^T} \rfloor, \tag{7}$$

where $\mathbf{C_r^q}$ and $\mathbf{C_c^q}$ are the corresponding matrixes for row and column transforms given by

$$\mathbf{C_r^q} = \frac{\mathbf{C}}{q_r} \quad , \quad \mathbf{C_c^q} = \frac{\mathbf{C}}{q_c} \quad , \quad q_r \times q_c = q.$$

To use the same coefficients for both row and column transforms, we should use the same quantization factor, i.e.,

$$q_r = q_c = \sqrt{q} \quad ,$$

and have $\mathbf{C_r^q}$=$\mathbf{C_c^q}$=$\mathbf{C^q}$. Equation (7) then becomes

$$\mathbf{Y^q} = \lfloor \mathbf{C^q X (C^q)^T} \rfloor. \tag{8}$$

3. IMPLEMENTATION OF THE QDCT WITHIN AN MPEG-2 FRAMEWORK

Until now, we have assumed that the coefficients have high precision, i.e. they are represented in double floating-point format. In a practical implementation where integer computations are required, the coefficients are scaled up to form the integer coefficients. At the end, the results are scaled down. Let's assume that the coefficients are scaled up to b bits integers. To make the rounding operation even simpler, we can go one step further and compute an approximation of (8) as follows

$$\mathbf{Y^q} \approx \frac{\mathbf{C_i^q X (C_i^q)^T}}{2^{2b}} \tag{9}$$

where each element of $\mathbf{C^q}$ is scaled up and rounded to the nearest integer to produce the corresponding element of $\mathbf{C_i^q}$, that is,

$$\mathbf{C_i^q} = \lfloor \mathbf{C^q} \times 2^b + 0.5 \rfloor.$$

Although a detailed analysis of this approximation error is still necessary, we could, in practice, obtain some useful results by just decreasing b from an initial very high value until the results deviate from the ideal case. For MPEG-2 video encoding, setting b to $b = 10$ yields virtually no loss in performance. Setting b to $b = 8$ introduces a slight error, but this value of b makes scaling extremely simple.

As the result of this rounding process, we now have a set of integer coefficients for each value of quantization scale. The number of coefficients in the set depends on the specific method used to implement the QDCT. The computation of QDCT is identical to the computation of DCT (only replacing the appropriate coefficients). For instance, if we use Chen's algorithm [3] to compute the QDCT, the numbers of coefficients that need to be stored in the set for each value of q is 7.

In MPEG-2 video encoding [1], there are two kinds of coding modes: intra and inter. Intra coded blocks are coded using a quantization matrix with different values for each position in the block. Thus, the proposed QDCT cannot be easily applied. However, inter coded blocks (which represent more than 80% of all coded blocks) are usually quantized using the same quantization factor for all coefficients in the block. In this work, we use the linear quantization scale for MPEG-2 encoding, which means that the quantization scale qs can take 31 values from 2,4,6,... up to 62. If we use Chen's algorithm for implementation of the QDCT, the number of coefficients that need to be stored in this case will be $31 \times 7 = 217$. The values of these coefficients are shown in Table 1 (these values are with b=10).

qs	QDCT coefficients						
2	256	355	301	201	71	334	139
4	181	251	213	142	50	237	98
6	148	205	174	116	41	193	80
8	128	178	151	101	35	167	69
...	...	...	...	...	...	...	...
60	47	65	55	37	13	61	25
62	46	64	54	36	13	60	25

Table 1: QDCT coefficients when using Chen's DCT implementation

To compare the performance in quality of the combined method to the traditional method (called DCT+Q in this paper), simulation experiments are carried out by replacing the DCT and quantization for inter coded blocks by the QDCT. Averages for all sequences coded at 3MBps are shown in Table 2.

Method	Average PSNR (dB)
DCT+Q (original TM5)	29.67
QDCT (10 bits scaling)	29.68
QDCT (8 bits scaling)	29.65

Table 2: Performance of QDCT compared to DCT+Q

The results show that the QDCT attains the same performance level as the traditional method, even with 8 bits of scaling.

4. COMPUTATIONAL COMPLEXITY

To compare the computational complexity between the traditional DCT+Q method and the QDCT, the specific numbers of computations for the implementation of each method using some well-known fast algorithms are shown in Table 3 and Table 4. These tables are adapted from [2]. Within them, 1D means that the transform is separable. M,A and S denote Multiplications, Additions and Shifts, respectively. We also neglect all the number of shifts used in the intermediate stages of the fast algorithms, as this is normally very small. The number of shifts shown in the tables accounts only for scaling.

One can see that the number of computations are identical for QDCT and DCT+Q when both are implemented using fast, scaled algorithms. The reason can be explained as follows. In the traditional DCT+Q method, if we assume that the computation of an 8×8 block scaled-DCT requires M, A and S operations, then for yielding the true DCT coefficients, the corresponding numbers of operations are M+64,A and S+64. After that, the computation of quantization requires another 64M and 64S. Thus, the total numbers of operations are M+128,A and S+128. However, in scaled DCT algorithms, scaling to the true DCT coefficients and quantization are always combined. Therefore, the total numbers of operations is reduced to M+64, A and S+64. On the QDCT side, the implementation of the QDCT using the same fast, scaled structure will require M,A and S operations to yield the quantized, scaled coefficients. Another scaling step is needed to compute the quantized, non-scaled coefficients. This leads to the same numbers of operations as in the DCT+Q case: M+64,A and S+64.

DCT and Q	1D			2D		
	M	A	S	M	A	S
1D Chen	24	26	8	320	416	64
1D LLN	19	29	8	240	464	64
2D Cho,Lee				160	466	64
1D Chen (scaled)	16	26	8	192	416	64
1D Winograd (scaled)	13	29	8	144	464	64
2D Feig,Win. (scaled)				118	462	64

Table 3: Computational complexity of DCT+Q implemented by various fast algorithms

QDCT	1D			2D		
	M	A	S	M	A	S
1D Chen	16	26	8	256	416	64
1D LLN	11	29	8	176	464	64
2D Cho,Lee				96	466	64
1D Chen (scaled)	16	26	8	192	416	64
1D Winograd (scaled)	13	29	8	144	464	64
2D Feig,Win. (scaled)				118	462	64

Table 4: Computational complexity of QDCT implemented by various fast algorithms

These tables also show that in terms of computational complexity, the QDCT is more efficient than the DCT+Q when it is implemented using a two-dimensional, non-scaled fast algorithm. Moreover, the QDCT also has an additional advantage: by embedding the quantization into the transform, the range of the quantized coefficients will decrease (instead of increase, like after DCT) in the computational process. This suggests a method to further reduce the number of computations, as described in the next section.

5. EARLY TERMINATION AND PREDICTION

5.1. Early Termination

Because the QDCT is the combination of a transform and quantization, it is expected that the output data (quantized coefficients) are more localized (i.e., having a more narrow, peaked distribution). In MPEG-2 inter block encoding, the input data represent motion compensated prediction error, which is centered narrowly around zero. Simulation results show that at a bitrate of 3MBps (which is the bitrate of our interest), 84% of all blocks are inter coded blocks and 80% of such blocks become zero valued after quantization. It would greatly save computations if we could detect or predict these blocks in the early computation stages.

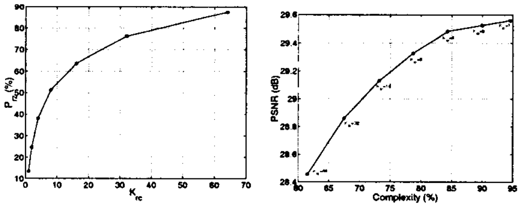
Indeed, if the computation of the QDCT is implemented on a row and column basis, an all-zero check can be made after the row transforms. If all transformed rows are zero, no further computations are necessary. This Early Termination technique can be described compactly by

```
begin Do 8 row transforms
  if block of 8 rows is zero
    then exit
    else Do 8 column transforms
```

where *exit* means a 8×8 blockof all zeros is returned. Although we now have an overhead for the comparison, this overhead is indeed rather small compared to the amount of computations for the QDCT. It's also not necessary to do the comparison on a coefficient-to-coefficient basis. For instance, if each coefficient is represented using 16 bit integer, on 32 or 64 bit CPU we can compare 2 or 4 coefficients in one step. Some other platforms might support even more efficient mechanisms for detecting an all-zero block.

It is easy to see that by applying Early Termination technique to the QDCT, the gain in computations is a function of the probability of the blocks being zero after the row transforms. This probability P_{rz}, in turn, is a function of the input data characteristics and the quantization factor. A high value of P_{rz} yields a low computational complexity for the implementation of the QDCT using the Early Termination technique, and vice versa. In MPEG-2 video encoding, this probability is rather small, as seen in Figure 1(a) below. One way to increase it is to weight more the quantization process embedded in the row transforms of the QDCT. This can be done by setting the value of q_r higher than the value of q_c, e.g. $K_{rc} = q_r/q_c > 1$. In this case, we have to maintain two sets of QDCT coefficients, one for the row transforms and the other for the column transforms. Figure 1(a) shows the relation between P_{rz} and K_{rc}. When K_{rc} increases, more coefficients will become zero after the row transforms and the method also becomes less precise. When $K_{rc} \to \infty$, the loss compared to the ideal case is approximately 3dB, but no computation is then necessary. This is the case where the decoder relies totally on motion compensated prediction.

The trade-off between computational complexity and PSNR is shown in Figure 1(b). In this figure, the computation gain, or the complexity (represented in %) is normalized as the ratio between the number of 8-point 1D-QDCT (row or column) actually computed and the number of 8-point 1D-QDCT if computed using standard row-column method (which is 16 times the number of coded blocks). By using this computation measure, we have an accurate evaluation of the computational reduction that is independent of any specific implementation.



(a) P_{rz} vs. K_{rc}

(b) PSNR vs. Complexity

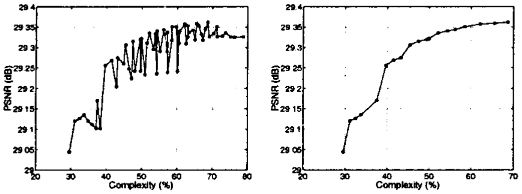
Figure 1: Trade-offs for Early termination

5.2. Early Prediction

To further reduce the computational complexity, the Early Termination technique can be generalized to become an Early Prediction technique as follows:

```
Do $N_r$ row transforms
if block of $N_r$ transformed rows are zero
  then exit
  else Do $8 - N_r$ row transforms for the remaining rows
      Do $N_c$ column transforms
      if block of $N_c$ transformed columns are zero
        then exit
        else Do $8 - N_c$ column transforms for the remaining columns
$(N_r, N_c = 1 \ldots 8)$
```

The Early Termination technique described in Section 5.1 can be considered as a special case of this method where $N_r = N_c = 8$. We now have more degrees of freedom, as all three parameters K_{rc}, N_r and N_c can be changed. To evaluate the trade-off, for each combination of K_{rc}, N_r and N_c, we simulate to obtain the corresponding PSNR and the normalized complexity. Then the results are grouped according to K_{rc} and sorted according to the complexity. For example, Figure 2(a) shows the simulated results corresponding to $K_{rc} = 8$, after these results are sorted in ascending order of the normalized complexity. A simple filter routine is used to remove all points that have PSNR value lower than any PSNR value of other points on their left side. The corresponding filtered version of the curve in Figure 2(a) is shown in Figure 2(b). Figure 3 shows the trade-offs corresponding to K_{rc}=2...32. The dashed curve represents the overall optimum trade-off. It is formed by mixing the data of all other curves, then sorting and filtering as desribed above. Each point on this curve represents a triplet (K_{rc}, N_r, N_c). Some of these triplets are shown in Table 5. The performance of the optimum trade-off in this case is apparently better than the one shown in Figure 1(b). This is reasonable because with N_r and N_c, we have two more degrees of freedom in choosing the optimal set of parameters.



(a) sorted

(b) sorted and filtered

Figure 2: Trade-offs for all values of N_r and N_c when $K_{rc} = 8$

PSNR(dB)	Compl.	K_{rc}	N_r	N_c
29.6	72	1	4	4
29.5	46	2	1	2
29.4	40	4	1	3
29.1	31	8	1	2
28.5	19	32	1	4

Table 5: Some values derived from the dashed curve in Figure 3

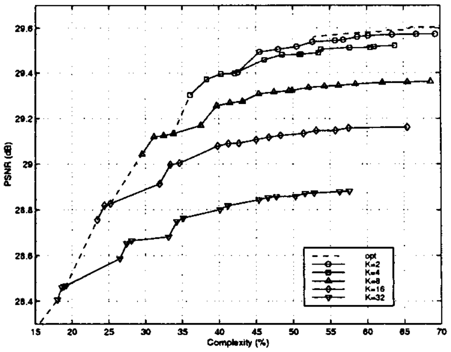


Figure 3: Trade-offs for various K_{rc}, N_r and N_c

It is clear that one can choose a specific triplet of (K_{rc},N_r,N_c) to minimize the computational complexity corresponding to a tolerable loss of quality, or from a fixed, constrained number of computational complexity, choose the right triplet to maximize the PSNR.

6. CONCLUSIONS

A new technique for the combination of DCT and quantization, namely Quantized Discrete Cosine Transform (QDCT), is presented. The QDCT implementation using integer coefficients can be used with general purpose CPU or DSP chips. In this case, it yields virtually no loss compared to the traditional method of separated DCT and quantization. For time-constrained applications where trade-offs between complexity and quality are desired, we introduce a new method for early prediction of zero blocks. Optimum parameters for this method can be obtained off-line, as demonstrated in this paper.

7. REFERENCES

[1] J.L.Mitchell,W.B.Pennebaker,C.E.Fogg,D.J.LeGall "MPEG Video Compression Standard" Chapman & Hall 1996.

[2] V.Baskaran,K.Konstantinides "Image and Video Compression Standards" Kluwer Academic Publisher 1997.

[3] K.R.Rao,P.Yip "Discrete Cosine Transform - Algorithms, Advantages, Appications" Academic Press 1990.

[4] B.G.Lee "A new algorithm to compute the dicrete cosine transform" IEEE Trans on ASSP 12/1984.

[5] W.B.Pennebaker,J.L.Mitchell "JPEG Still image data compression standard" Van Nostrand Reinhold 1993.