KUFA JOURNAL OF ENGINEERING
ISSN 2071-5528
PRINTED IN IRAQ

Volume 3, Number 2, pp.1-16, 2012

Low Complexity Embedded Image Compression Algorithm using Subband Coding of the DCT Coefficients

Ali Kadhim Jaber, Ph. D. Information Technology,

Department of Electrical Engineering, College of Engineering, University of Kufa

(Received:22/5/2011 ;Accepted :11/10/2011)

Abstract

This paper proposes a new algorithm termed Quad Tree-Discrete Cosine Transform (QT-DCT). QT-DCT is a low complexity algorithm that combines the good energy compaction ability and the simplicity of the DCT used by the Joint Photographic Expert Group (JPEG) image compression standard, and the global and embedding properties of Subband Coding (SBC) systems. The proposed QT-DCT reorders the DCT coefficients into subbands. Then these subbands are coded using quad-tree decomposition method. The main contribution of the proposed QT-DCT algorithm is that it has better PSNR and lower complexity than JPEG image compression standard and some existing algorithms that have about the same complexity. In addition, QT-DCT produces an embedded compressed bit-stream which is able to achieve any compression bit-rate very easily. This is a very desirable feature for the modern requirements represented by the Internet.

Keywords: Image Compression, JPEG, DCT, Subband Coding, Embedded Coding.

خوارزمية ضغط صور قليلة التعقيد ومتراكبة باستخدام نظام الترميز الجزئي للحزمة الترددية

الخلاصة

في هذا البحث تم تصميم خوارزمية جديدة لضغط الصور. النظام (QT-DCT) المقترح يجمع بين محاسن (DCT) المتمثلة بالبساطة (قلة وقت المعالجة) والقابلية الجيدة لتجميع الطاقة و محاسن نظام الترميز الجزئي للحزمة الترددية (Subband Coding) المتمثلة بالشمولية و قابليتها على انتاج صور مضغوطة متراكبة (embedded) لها امكانية استرجاع بعدة معدلات ضغط (compression bit-rates). يقوم (QT-DCT) بأعادة ترتيب معاملات (DCT) الى هيئة حزم جزئية (Subbands) ومن ثم يرمز هذه الحزم بأستخدام تقسيم الشجرة الرباعي (quad-tree decomposition). من اهم ميزات هذا النظام المقترح بأنه ذات كفاءة اعلى ووقت معالجة اقل مقارنة بالنظام القياسي لضغط الصور(JPEG) وبعض الأنظمة الأخرى الموجودة بألأضافة الى انه من النوع المتراكب وهذه الخاصية مهمة جدا لانظمة الضغط الحديثة وخصوصا عند ارسال الصور عبر الأنترنت.

1. Introduction

Most existing high performance image coders in applications are transform based coders because they provide better performance for a fixed complexity as compared to the spatial methods. The two popular transforms for image compression are the Discrete Cosine Transform (DCT) and Subband Coding (SBC) such as the Discrete Wavelet Transform (DWT) based systems. The DCT yields nearly optimal energy compaction. However, since the cosine basis functions are non-local, it is not practical to apply the DCT to an entire image because of spatial ringing artifact when high frequency coefficients are quantized. To localize the cosine bases, the DCT is usually implemented as block-based transform by which the image is normally divided into a number of non-overlapping blocks (e.g. 8 × 8 pixels) and the 2-Diomentional DCT (2-D DCT) is applied to these blocks individually. In this way, the high frequency ringing artifacts are reduced. But unfortunately, at low bit rates, block-based DCT suffers from the blocking artifacts, i.e., discontinuities at the block boundaries as shown in Figure (1) for "Lena" test image. In addition, block-based DCT is unable to exploit the correlations between the DCT coefficients across block boundaries. Nevertheless, Block-based DCT is the basis of many international multimedia compression standards such as Joint Photographic Expert Group (JPEG) for still images compression and the Motion PEG (MPEG) for video compression. Low computational complexity and reasonable compression performance are the main reasons for the popularity of the block-based DCT systems [Salomon 04], [Shi 08].

(a) **(b)**

Figure (1): (a) Original "Lena Image"; (b) Reconstructed Lena with JPEG at 0.25 bpp, to show blocking artifacts

On the other hand, Subband Coding (SBC) systems can deal with the entire input image and avoid blocking artifacts by employing overlapped basis functions. The basic idea of SBC is to decompose the input image into different frequency bands by applying frequency-selective filter-bank which consists of a collection of filters. Each filter has different center frequency. Therefore, the image is decomposed into different frequency bands termed subbands. Each subband may have different characteristics. Coding technique best suited to each subband can be used to improve the compression performance. Unfortunately, SBC has higher computation complexity than that of the DCT: the most efficient algorithm for the (8 × 8) two-dimensional (2-D) DCT requires only 54 multiplications, while the complexity of the wavelet SBC depends on the length of the wavelet filters, which is at least one multiplication per coefficient [Xiong 99].

One additional drawback of JPEG encoding is the difficulty of the Rate Distortion Control (RDC). RDC involves obtaining the best image quality (lowest distortion) for a given compression data rate (DR). In JPEG, RDC consists of choosing a quality factor which serves as a scaling factor of the standard quantization tables. A problem with such approach is that there is no guarantee that the scaling factor will give the target data rate exactly. Therefore, the encoder must compress the image multiple times. Each time with different scaling factor until the target compression DR is attained. Evidently, this solution increases the complexity of the algorithm and the associated delay time. Another solution is to produce and store multiple bit-streams that are compressed at different data rates. Evidently, this will eliminate the need to re-encoding but at the same time it increases the memory requirements and complicates the memory management due to manipulating multiple bit-streams [Salomon 04].

The RDC problem can be solved by using embedded coding technique. With embedded coding, the resulting bit-stream can be decoded at different data rates. This is achieved by arranging the information in the bit-stream according to its importance in decreasing order, i.e. the most important information (the information that has the highest distortion reduction) is put at the beginning of the bit-stream, followed by the less important, and so on. In this way RDC can be easily achieved by simply truncating the bit-stream without the need of repeating the compression process and eliminating the need for a buffer control method when fitting the data to a certain given data rate [Shi 08], [Vetterli 95].

Several works are presented in the literatures that produced an embedded bit-stream for the DCT-based systems. The algorithms of [Tran 99], [Van 00], and [Xiong 96] provided about the same efficiency as JPEG but at lower complexity. [Samai 06] studied the performance of the algorithm of [Xiong 96] with variable DCT block size. [Chengjie 02] presented the Embedded-Context-based Entropy coding of Block transform coefficients (E-CEB) algorithm. E-CEB reorders the DCT blocks into subbands. Then it codes the subband's coefficients bit-planes efficiently using binary arithmetic coder with high-order space-frequency context modeling. E-CEB has good performance but at the expenses of highly increased complexity since the complexity of the arithmetic coder is about 4 times more than Huffman coder used in JPEG. This means that E-CEB is at least 4 times slower than JPEG. The high complexity of E-CEB coding method looses the low complexity benefit gained by using the DCT instead of SBC. That is, the complexity reduction attained by using the DCT is lost by the complexity increment due to using the arithmetic coder with context modeling.

2. The Proposed Quad Tree-DCT (QT-DCT) Algorithm

The proposed QT-DCT algorithm combines the excellent energy compaction ability and simplicity of the DCT and the global and the embedding properties of the SBC systems. The proposed algorithm outperforms standard JPEG coding in terms of Peak Signal to Noise Ratio (PSNR) versus DR. In addition, the embedded feature of the coder allows the encoding and/or decoding to achieve exact compression DR with the best possible image quality. Furthermore, the proposed QT-DCT algorithm has less complexity than the original JPEG because it is based on the zero-tree coding approach which is simpler than the Huffman or the arithmetic coding used by JPEG [Pearlman 01]. The QT-DCT algorithm consists of two main stages: transformations and subbands construction stage, and the embedded coding stage.

2.1 Transformations and Subbands Construction

2.1.1 DCT Transformation

In this step the image is subdivided into non-overlapping blocks of size 8 × 8 pixels each. Then a 2-D DCT is performed on each block, processing them in raster order from left to right, top to bottom. Figure (2.a) depicts an image of size 8 × 8 pixels that is transformed using 4 × 4 DCT blocks. It should be noted that the resulting DCT coefficients are floating-point numbers.

2.1.2 Subband Construction

An M-point DCT block transform can be interpreted as an M-band filter bank whose filters are simply the transform's basis functions [Vetterli 95], [Chengjie 02]. So, the DCT coefficients can be rearranged into a subband structure (according to their frequencies). Subband *ij* collects all coefficients at the position *ij* from every DCT block. These coefficients represent the same frequency component of the entire image. For instance, subband 00 gathers the coefficients at position 00 (the DC coefficients) from all DCT blocks; subband 01 gathers the coefficients at position 01; and so on. Figure (2.b) illustrates the subbands formation of the image in Figure (2.a).

A_{00}	A_{01}	A_{02}	A_{03}	B_{00}	B_{01}	B_{02}	B_{03}
A_{10}	A_{11}	A_{12}	A_{13}	B_{10}	B_{11}	B_{12}	B_{13}
A_{20}	A_{21}	A_{22}	A_{23}	B_{20}	B_{21}	B_{22}	B_{23}
A_{30}	A_{31}	A_{32}	A_{33}	B_{30}	B_{31}	B_{32}	B_{33}
C_{00}	C_{01}	C_{02}	C_{03}	D_{00}	D_{01}	D_{02}	D_{03}
C_{10}	C_{11}	C_{12}	C_{13}	D_{10}	D_{11}	D_{12}	D_{13}
C_{20}	C_{21}	C_{22}	C_{23}	D_{20}	D_{21}	D_{22}	D_{23}
C_{30}	C_{31}	C_{32}	C_{33}	D_{30}	D_{31}	D_{32}	D_{33}

(a) 8 × 8 image transformed using 4 × 4 DCT blocks

A_{00}	B_{00}	A_{01}	B_{01}	A_{02}	B_{02}	A_{03}	B_{03}
C_{00}	D_{00}	C_{01}	D_{01}	C_{02}	D_{02}	C_{03}	D_{03}
A_{10}	B_{10}	A_{11}	B_{11}	A_{12}	B_{12}	A_{13}	B_{13}
C_{10}	D_{10}	C_{11}	D_{11}	C_{12}	D_{12}	C_{13}	D_{13}
A_{20}	B_{20}	A_{21}	B_{21}	A_{22}	B_{22}	A_{23}	B_{23}
C_{20}	D_{20}	C_{21}	D_{21}	C_{22}	D_{22}	C_{23}	D_{23}
A_{30}	B_{30}	A_{31}	B_{31}	A_{32}	B_{32}	A_{33}	B_{33}
C_{30}	D_{30}	C_{31}	D_{31}	C_{32}	D_{32}	C_{33}	D_{33}

(b) Reordered DCT blocks into subbands structure

Figure (2): The two different representations of DCT coefficients. (a) The block representation and (b) The subband representation

Figure (3) shows the block and subband representation of 8 × 8 DCT coefficients of the "Lena" image. Spatial features are much more visually obvious in subbands than in blocks. The lowest frequency subband, which is obtained by grouping the DC coefficients of all DCT blocks,

represents a thumbnail image (an image that contains all the details of the original image but at widely reduced size). Each of the other subbands contains a part of the global information of the image.

(a) DCT block representation **(b) Subband representation**

Figure (3): (a) DCT block representation and (b) Subband representation for "Lena" Image.

2.2 Embedded Coding of the Subbands

This stage constructs an embedded bit-stream for the image using bit-plane coding, whereby the magnitude bits of the DCT coefficients are coded from most significant bit (MSB) to least significant bit (LSB). That is, the image is coded bit-plane by bit-plane rather than coefficient by coefficient. This is because the MSBs have higher information contents than the LSBs and thus they tend to reduce the distortion most. So, an exact data rate (with the best performance) can be easily attained by simply truncating the bit-stream without the need to use an explicit rate distortion control (RDC) process which leads to complexity reduction as compared to the non-embedded compression systems.

2.2.1 Bit-plane Coding

In order to implement bit-plane coding, the floating-point DCT coefficients are initially quantized by rounding them to the nearest integers (e.g. +75.25 ≡ +75). Then, every quantized DCT coefficient $\mathbf{q_{ij}}$ is represented by the sign-magnitude format using fixed number of bits (e.g. 16 bits) where the leftmost bit is the sign bit (0 for positive and 1 for negative coefficient), and the

remaining bits are the magnitude bits. Denote the magnitude bit and the sign bit of $\mathbf{q_{ij}}$ at bit-plane **p** by $\mathbf{q_{ij}^p}$ and $\mathbf{q_{ij}^s}$ respectively. For each coefficient, we call its first non-zero bit (starting from MSB) the First Significant Bit (FSB). The bits of a coefficient prior to the FSB are referred to as the Zero bits (ZBs), while the the bits after the FSB are called Refinement bits (RBs). A coefficient $\mathbf{q_{ij}}$ is considered significant (SG) with respect to bit-plane **p** if its FSB is found, i.e., if $\mathbf{q_{ij}^p}$ is the FSB; otherwise $\mathbf{q_{ij}}$ is Insignificant (IS). Similarly, a set of coefficients ***S*** is considered SG if ***S*** contains one or more SG coefficients; otherwise ***S*** is IS.

Due to the energy compaction property of the DCT, some of the most significant bit-planes may have ZBs only. So, there is no need to code them. The coding starts at the maximum bit-plane (P_{max}) which is the bit-plane that has at least one SG coefficient in the entire DCT image. $\mathbf{P_{max}}$ depends on the maximum coefficient in the DCT image and it is given by [Pearlman 01], [Salomon 04]:

$$\mathbf{P_{max} = \lfloor \log_2(\max | q_{ij} |) \rfloor} \qquad \mathbf{(1)}$$

where $\lfloor x \rfloor$ is the largest integer smaller than or equal to x. $\mathbf{P_{max}}$ should be signaled to the decoder in order to start decoding at bit-plane $\mathbf{P_{max}}$.

Obviously, coding the individual bits one by one is not efficient. An efficient bit-plane coder should exploit the statistical properties of individual bits within and/or across the image subbands to improve the coding efficiency. This goal may be achieved by using entropy coder, such as Huffman or arithmetic coder. Unfortunately, these methods are complex and slow. A simpler method is the zero-tree coding that uses a hierarchical set partitioning process to split off the SG coefficients, while maintaining areas of IS coefficients. In this way, a large region of zero bits (that represent IS DCT coefficients) can be coded by one symbol (one bit). This allows for the coding of a large number of IS coefficients by only coding the location of the root to which the entire set of zero bit coefficients is related [4]. QT-DCT uses the Quad-Tree Decomposition (QTD) as a set partitioning rule for the SG sets. The QTD procedure is very simple because it needs only a binary search for the SG pixels [Islam 99].

Like other set partitioning schemes [Pearlman 01], QT-DCT uses three lists as follows:

- List of Insignificant Pixels (LIP):– stores the IS pixels that belong to SG sets.

- List of Significant Pixels (LSP):– stores pixels that are found to be SG in previous bit-plane passes.
- List of Insignificant Sets (LIS):– stores all the sets (with at least 2×2 pixels) that are IS but belong to larger SG sets.

A pixel in LIP and LSP is represented by its (i, j) coordinates, while a set in LIS is represented by the (i, j) coordinates of its top-left pixel and by its size *z*. LIP and LSP are initialized as empty lists and LIS is initialized by all image subbands.

Every coding pass –except the first–consists of three sub-passes. In order, these sub-passes are termed the Test Insignificant Pixels (TIP), Test Insignificant Sets (TIS), and the Refinement (REF) sub-passes respectively. In the TIP sub-pass, each pixel $\mathbf{q_{ij}}$ in LIP is coded by sending its $\mathbf{p^{th}}$ bit, $\mathbf{q_{ij}^{p}}$, directly to the bit-stream. And, if the pixel is found to be SG (with respect to the current bit-plane) its sign bit, $\mathbf{q_{ij}^{s}}$, is coded and its (i, j) coordinates are moved to end of LSP. In the TIS sub-pass, each set in LIS –except the sets that will be added in the current pass– is removed from LIS and passed to QTD procedure (described in the next section) for encoding. Finally, in the REF sub-pass, all pixels of LSP which are found SG in the previous passes, i.e., except those added in the current pass are refined by sending their $\mathbf{p^{th}}$ RBs directly to the bit-stream.

2.2.2 QTD Procedure

As mentioned above, each set ***S*** in LIS is passed to the QTD Procedure for coding. The QTD process starts by testing the input set ***S*** for significance. If ***S*** is still IS a 0 is sent to the bit-stream and it is added back to LIS to be tested in the next bit-planes. If ***S*** is SG, a 1 is sent to the bit-stream, and it is partitioned into four children subsets each having one-fourth the size of the parent set ***S***. Each one of these subsets is in turn treated as a set of type ***S*** and recursively passed to the QTD procedure to be processed in the same way as the parent set ***S***, i.e. the SG sets are partitioned and the IS sets are added to LIS. This recursive quad-tree partitioning of the SG sets continues until the pixel-level is reached. Each pixel is then coded by sending its $\mathbf{p^{th}}$ magnitude bit, $\mathbf{q_{ij}^{p}}$, to the bit-stream, and if the pixel is found to be SG, its sign bit, $\mathbf{q_{ij}^{s}}$, is also sent to the bit-stream and its (i, j) coordinates are appended to the end of LSP. The IS pixels are added to LIP to be tested in the next bit-planes.

Figure (4) shows an example of the QTD process for an (8 × 8) pixels set with its top-left pixel is at (0, 0). Define $\mathbf{S_{ij}^{z}}$ as a set $\boldsymbol{S}$ with z × z pixels and its top-left pixel is at (i, j) coordinates (which is considered as the root of the set). So, the input set $\boldsymbol{S}$ is represented as $\mathbf{S_{00}^{8}}$ because its top-left pixel is at (0, 0) and it has 8×8 pixels. $\mathbf{S_{00}^{8}}$ is SG because it has two SG pixels at (0, 3), and (4, 5). So, 1 is sent to the bit-stream and $\mathbf{S_{00}^{8}}$ is quadrisected (i.e., divided into four parts) to obtain four subsets ($\mathbf{S_{00}^{4}}, \mathbf{S_{04}^{4}}, \mathbf{S_{40}^{4}}$, and $\mathbf{S_{44}^{4}}$) as shown in Figure (4.a) for the first stage of QTD. In the second stage of QTD, each one of these subsets is recursively passed to the QTD in order to be processed in the same way as the first stage. The sets $\mathbf{S_{00}^{4}}$ and $\mathbf{S_{44}^{4}}$ are SG, so a 1 is sent to the bit-stream for each set. Then the set $\mathbf{S_{00}^{4}}$ is quadrisected to the subsets ($\mathbf{S_{00}^{2}}, \mathbf{S_{02}^{2}}, \mathbf{S_{20}^{2}}$, and $\mathbf{S_{22}^{2}}$) and the set $\mathbf{S_{44}^{4}}$ is quadrisected to the subsets ($\mathbf{S_{44}^{2}}, \mathbf{S_{46}^{2}}, \mathbf{S_{64}^{2}}$, and $\mathbf{S_{66}^{2}}$) as shown as shown in Figure (4.b). On the other hand, the sets $\mathbf{S_{04}^{4}}$ and $\mathbf{S_{40}^{4}}$ are IS, so a 0 is sent for each set and they are transferred to LIS in order to be tested in the next bit-plane. In the third stage of QTD, the sets $\mathbf{S_{02}^{2}}$ and $\mathbf{S_{44}^{2}}$ are quadrisected and 1 is sent to the bit-stream for each set because they are SG as shown as shown in Figure (4.c). The other six sets are IS so a 0 is sent for each set and they are transferred to LIS in order to be tested in the next bit-plane. It should be noted that the pixel level is reached at the third QTD stage, so the individual pixels of the SG sets must be coded.

	0	1	2	3	4	5	6	7
0	0	0	0	**1**	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	**1**	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

(a) The first stage of the QTD procedure

	0	1	2	3	4	5	6	7
0	0	0	0	**1**	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	**1**	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

(b) The second stage of the QTD procedure

	0	1	2	3	4	5	6	7
0	0	0	0	**1**	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	**1**	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

(c) The third stage of the QTD procedure

Figure (4): QTD procedure for (8× 8) pixels significant set

2.2.3 QT-DCT Decoding

The decoder uses the same mechanism as the encoder. It receives significance test results from the bit-stream and builds up the same list structure during the execution of the algorithm. Hence, it is able to follow the same execution paths for the significance tests of the different sets, and reconstructs the image progressively as the algorithm proceeds. It is easy to see that with this scheme, the decoder has lower computational complexity than the coder. This is because the decoder doesn't need to scan the set's coefficients to see if the set is SG or not: at any execution point, when the decoder receives '1', the corresponding set is SG; otherwise it is IS. Thus the whole set can be bypassed. This is very consistent with scalable compression schemes as the image is compressed once and may be decompressed many times.

3. Experimental Results and Discussion

The proposed QT-DCT algorithm is evaluated and tested using C++. Results of [Van 00] and E-CEB [Chengjie 02] algorithms are also included for the purpose of comparison. The algorithm of [Van 00] is selected because it has about the same complexity as the proposed QT-DCT and it has the best performance among the existing algorithms that have about the same complexity as the QT-DCT. The algorithm of [Chengjie 02] is selected in spite of its higher complexity in order to demonstrate the speed advantage of QT-DCT. The performance is evaluated using the grayscale 8 bit per pixel (bpp), 512 × 512 pixels "Lena" (shown in Figure (1.a)), "Goldhill", and "Barbara" test images (shown in Figure (5)).

(a) Goldhill

(b) Barbara

Figure (5): Test images

The performance is measured by the Peak Signal to Noise Ratio (PSNR) versus data rate (DR). For grayscale image with 8bpp, PSNR is defined as [1]:

$$\mathbf{PSNR = 10\log_{10}\frac{255^2}{MSE}\quad (dB)} \tag{2}$$

Where MSE is Mean-Squared Error between the original and the reconstructed images. it is defined as:

$$\mathbf{MSE = \frac{1}{M \times N}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(X[i,j] - \hat{X}[i,j])^2} \tag{3}$$

where **X** is the original image, $\hat{\mathbf{X}}$ is the reconstructed image, and **M×N** is the image size (number of pixels). Evidently, smaller MSE and larger PSNR values correspond to lower levels of distortion. The data rate (DR) of the compressed image is measured by the average number of bits per image pixel, where this average is taken over the entire image. DR is given by:

$$\mathbf{DR} = \frac{\mathbf{size\ of\ compressed\ bit\text{-}stream\ (bits)}}{\mathbf{number\ of\ pixels\ in\ image}} \quad \mathbf{bit\ per\ pixel\ (bpp)} \tag{4}$$

Another commonly used metric is the compression ratio (CR), which is ratio of the size of the original image to the size of the compressed image [1]. It is defined as:

$$\mathbf{CR} = \frac{\mathbf{compressed\ image\ size\ (bits)}}{\mathbf{original\ image\ size\ (bits)}} \tag{5}$$

The performance of a lossy compression system may be described by its rate-distortion (R-D) characteristic, representing the potential trade-off between data rate and the distortion associated with the lossy representation. Obviously, for any DR or CR, the highest possible PSNR is desired and for any PSNR the highest possible CR or the lowest possible DR is desired.

Table (1): PSNR vs. DR for Lena, Barbara, and Goldhill images

		PSNR (dB)						
		Lena			**Goldhill**		**Barbara**	
DR (bpp)	**CR**	**[Van 00]**	**E-CEB**	**QT-DCT**	**E-CEB**	**QT-DCT**	**E-CEB**	**QT-DCT**
0.0625	**1:128**	-	26.82	**26.07 (-0.75)**	25.96	**25.37 (-0.59)**	22.73	**23.20 (+0.47)**
0.125	**1:64**	27.8	29.83	**28.58 (-1.25)**	27.99	**27.36 (-0.63)**	24.99	**24.97 (-0.02)**
0.25	**1:32**	30.5	33.16	**31.79 (-1.37)**	30.23	**29.47 (-0.76)**	27.83	**27.58 (-0.25)**
0.5	**1:16**	34.2	36.63	**35.42 (-1.21)**	32.97	**32.10 (-0.87)**	31.91	**31.21 (-0.70)**
1	**1:8**	37.3	40.08	**39.08 (-1.00)**	36.51	**35.84 (-0.67)**	36.98	**36.37 (-0.61)**

Table (1) shows the PSNR versus DR and CR of [Van 00], E-CEB and proposed QT-DCT algorithms. As it is clearly shown for "Lena", the proposed QT-DCT is better than the algorithm of [Van 00] that has about the same complexity as QT-DCT. On the other E-CEB has better PSNR than QT-DCT. For "Lena", QT-DCT has the maximum drop in PSNR (1.37 dB). For "Goldhill", which is more complex than "Lena", the PSNR drop is lower (0.87 dB). Finally, for "Barbara" image (which is more complex than "Lena" and "Goldhill" because it has much high frequency contents), the QT-DCT is very close to E-CEB especially at low data rates. This means that QT-DCT is more efficient for complex images than for simple images.

As previously mentioned, E-CEB is based on arithmetic coder which is about 4 times slower than the Huffman coder used with JPEG [Pearlman 01]. As QT-DCT has lower complexity than JPEG, E-CEB is in turn has higher complexity than QT-DCT. In order to demonstrate this, the E-CEB is implemented and its coding time (time required by the algorithm to encode the image at given data rate) is depicted in Figure (6) together with that of QT-DCT for "Lena" image. The speed advantage of QT-DCT over E-CEB is obvious; the QT-DCT runs faster than E-CEB by about (7-18) times. That is, the price paid for the PSNR superiority of E-CEB over QT-DCT is its higher complexity. This result is expected as the quad-tree decomposition method used by QT-DCT is much simpler than the arithmetic coding used by E-CEB.

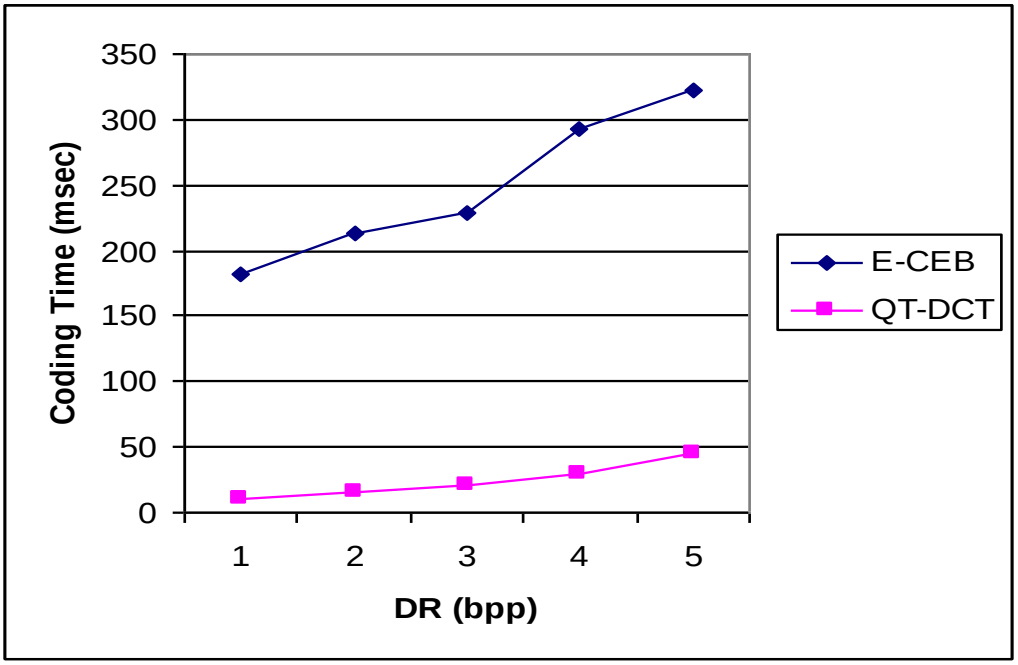

Figure (6): Coding time vs. DR of QT-DCT and E-CEB for "Lena" image

From Table (1) and Figure (6), we can deduce that E-CEB has higher PSNR and QT-DCT has lower complexity. So in order to give a fair comparison between the two algorithms, the PSNR to complexity (CX) ratio (PSNR/CX) is calculated against DR. CX is represented by the coding time given in Figure (6). Evidently a higher PSNR/CX is preferred as long as the PSNR is reasonable [Ordentlich 98]. Figure (7) gives PSNR/CX vs. DR of E-CEB and QT-DCT for "Lena" image that has the maximum PSNR drop. The PSNR/CX superiority of the proposed QT-DCT over E-CEB is clearly shown in the figure. That is, in spite of its lower PSNR, the proposed QT-DCT algorithm has higher PSNR/CX than E-CEB due to its much lower complexity.

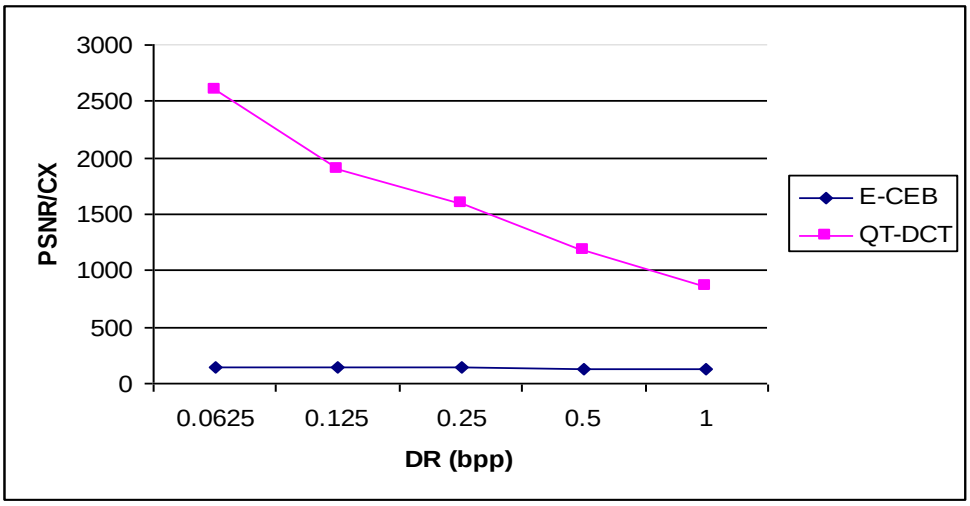

Figure (7): PSNR/CX ratio vs. DR of QT-DCT and E-CEB for "Lena" image

4. Conclusion

In this paper, we presented a very simple and efficient image compression algorithm. As practically demonstrated, the proposed QT-DCT algorithm provides better performance as compared to algorithms that have nearly the same complexity. On the other hand, compared to the more complex E-CEB algorithm, QT-DCT has slightly lower performance but it runs much faster and it has better performance to complexity ratio. Knowing that most people hate to wait a long time for data transfers, a faster algorithm is preferable as long as it has acceptable performance. In addition, the speed advantage of the proposed algorithm make it is very suitable for real-time applications such as video transmission where compression speed is more important than efficiency. Furthermore, a fast algorithm requires short processing time and consequently it consumes less energy. This means that QT-DCT can be used with limited power devices such as Mobile phones, Digital Cameras, etc. to preserve the life of the device's battery. Finally, and as mentioned previously, the DC subband (see Figure (3.b)) represents a thumbnail for the entire image or video

frame. Thumbnails are very useful for fast image and video browsing, as only a rough approximation of the image or video is sufficient for deciding whether the image needs to be decoded in full or not. Thus, the proposed QT-DCT can be used with Web image and video browsers for fast searching.

The only limitation of QT-DCT is its high memory requirements due to using the lists to store the coordinates of the image pixels. The total size of these lists is nearly equal to the size of the image. For example, for an image with 512 × 512 pixels, QT-DCT needs about 512×512 = 262 KByte of memory for the lists. However, with current technology, a RAM of 2 GB or more is affordable even with very small devices. Therefore, the high memory requirement of the QT-DCT is no longer a serious problem.

5. References

[Chengjie 02] Chengjie Tu et al. **"Context-Based Entropy Coding of Block Transform Coefficients for Image Compression,"** IEEE Trans. Image Processing, Vol. 11, No. 11, pp. 1271-1283, 2002.

[Islam 99] Islam A. et al. **"An Embedded and Efficient Low-Complexity Hierarchical Image Coder,"** Visual Comm. and Image Processing'99, Proceedings of SPIE, Vol. 3653, pp. 294-305, 1999.

[Ordentlich 98] Ordentlich E. et al. "**A Low-Complexity Modeling Approach for Embedded Coding of Wavelet Coefficients**," Proc. IEEE Data Compression Conf. (Snowbird), pp. 408-417, March, 1998.

[Pearlman 01] Pearlman W. A. "**Trends of Tree-Based, Set-Partitioning Compression Techniques in Still and Moving Image Systems**," Proceedings Picture Coding Symposium (PCS-2001), Seoul, Korea, 25-27, pp. 1-8, April, 2001.

[Salomon 04] Salomon D. "**Data Compression: the Complete Reference**," 3rd edition, Springer-Verlag Press, 2004.

[Samai 06] Samai et al. "**Image Compression via Embedded Coder in the Transform Domain**", Asian Journal of Information technology, Vol. 5, No. 6, pp. 633-639, 2006.

[Shi 08] Shi Y. Q. et al. "**Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and standards**," 2nd edition, CRC Press, 2008.

[Tran 99] Tran T. D. et al. **"A Progressive Transmission Image Coder using Linear Phase Uniform Filter-Banks as Block Transforms,"** IEEE Trans. Image Processing, vol. 8, Nov, pp. 1493-1507, 1999.

[Van 00] Van R. J. et al. **"Low-complexity Scalable DCT Image Compression,"** In Proceedings of International Conference on Image Processing, Vol. 3, IEEE, pp. 837-840, 2000.

[Vetterli 95] Vetterli M. et al. "**Wavelets and Subband Coding**," Prentice-Hall, New Jersey, 1st edition, 1995.

[Xiong 96] Xiong Z. et al. **"A DCT-Based Embedded Image Coder,"** IEEE Signal Processing Letters, vol. 3, no. 11, pp. 289-290, 1996.

[Xiong 99] Xiong Z. et al. **"A comparative study of DCT and wavelet-based image coding,"** IEEE Trans. Circuits & Systems for Video Technology, Vol. 9, No.5, pp. 692-695, Aug. 1999.