ADCTC: ADVANCED DCT-BASED IMAGE CODER

Nikolay Ponomarenko[1], Vladimir Lukin[1],
Karen Egiazarian[2] and Jaakko Astola[2]

[1]Department of Transmitters, Receivers and Signal Processing, National Aerospace University, Chkalova Str. 17, 61070, Kharkov, Ukraine, Telephone/Fax: +38 (057) 3151186, E-mails: uagames@mail.ru, lukin@xai.kharkov.ua
[2]Tampere University of Technology, Signal Processing Department, P. O. Box 553, FIN-33101, Tampere, Finland, Telephone: +358 3 3115 2923, +358 3 3115 3860, Fax: +358 3 3115 3087, E-mails: karen@cs.tut.fi, jta@cs.tut.fi

ABSTRACT

An Advanced DCT based image Coder (ADCTC) is proposed in this paper. It uses partition schemes, i.e. variable size DCT transforms of image blocks, coding of numbers of significant bits (NOSB), context modeling and coding of signs of DCT coefficients. Advantages of ADCTC in comparison to other recently proposed DCT and wavelet based coders including standards JPEG and JPEG2000 are demonstrated.

1. INTRODUCTION

Discrete cosine transform (DCT) [1] together with lapped transforms (LT) [2] and discrete wavelet transforms (DWT) [3] are widely used in many image coders, such as JPEG [4], SPIHT [5], EBCOT [6], JPEG2000 [7], etc.

In [8], authors presented a DCT-based image coder AGU-MHV utilizing modified horizontal-vertical partition scheme (that adaptively divides an image into non-equal size blocks), variable size DCT transforms and sophisticated bit-plane coding of DCT coefficients. This coder demonstrates superior performance comparing to other state-of-the-art wavelet based image coders, including SPIHT and JPEG2000. One of the bottlenecks of AGU-MHV coder is a relatively high computational complexity of both partitioning of an image into blocks and bit-plane coding of transform coefficients.

In this paper another advanced DCT-based image coder (ADCTC) based on partition schemes is proposed. It has much lower computational complexity than AGU-MHV demonstrating a relatively similar performance. ADCTC uses a simple cost function to find best partition scheme, faster coding of DCT coefficients, and simple method for coding signs of non-zero DCT coefficients. Note that the task of sign coding for DCT and DWT coefficients is quite actual [9-11] since signs often occupy more than 20% of total data of compressed images [10]. However, existed methods for sign prediction and coding are slow and not effective for large sizes of image blocks [10]. Therefore, in this paper we propose a novel, fast and simple method for sign coding that allows reducing the volume of data that relate to signs by about 5% (i.e., we additionally save about 1% of compressed data).

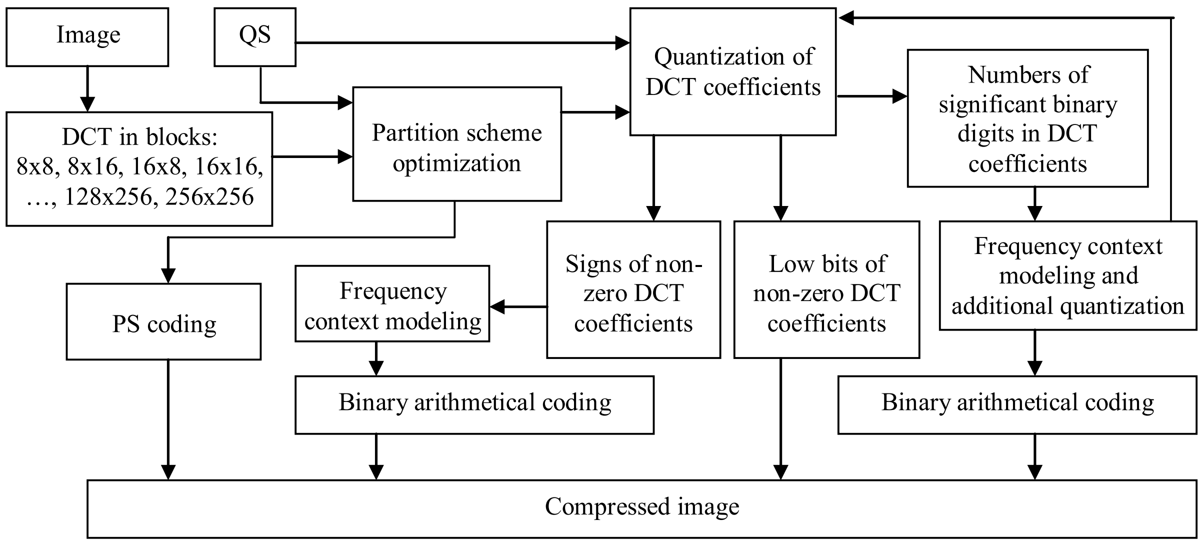
Image
QS
Quantization of DCT coefficients
Numbers of significant binary digits in DCT coefficients
DCT in blocks: 8x8, 8x16, 16x8, 16x16, …, 128x256, 256x256
Partition scheme optimization
PS coding
Frequency context modeling
Signs of non-zero DCT coefficients
Low bits of non-zero DCT coefficients
Frequency context modeling and additional quantization
Binary arithmetical coding
Binary arithmetical coding
Compressed image
Fig. 1. Flow-chart of image compression in the proposed coder ADCTC

This paper is organized as follows. Section 2 describes basic principles of operation of the coder ADCTC. Section 3 provides comparison results for the standard test set of grayscale images such as Lenna, Barbara, etc. Section 4 deals with more detailed comparison of performance for the coders JPEG, JPEG2000, and ADCTC for an extended set of 16 images of standard size 512x512 pixels. In Section 5, the same coders are analyzed for the set of 8 high resolution natural images (2288x1712 pixels) and image from a set of JPEG2000. ADCTC coder is publically available in [12].

2. DESCRIPTION OF THE PROPOSED CODER

Fig. 1 presents the flow-chart of gray-scale still image compression by ADCTC. Image decoding is carried out in inverse order. After decoding, fast post-filtering for blocking artifact removal (deblocking) is performed as in [8, 13, 14].

Let us describe a process of image compression by ADCTC. At the very beginning, optimization of the modified horizontal-vertical partition scheme (PS) [8] is carried out. Image is divided into blocks of possible sizes: 8x8, 8x16, 16x8, 16x16, 16x32, 32x16, 32x32, 32x64, 64x32, 64x64, 64x128, 128x64, 128x128, 128x256, 256x128, and 256x256 pixels. Moreover, for images that have both sides less than 1024 pixels, the maximal size of blocks we have used were 64x64 pixels. If images are larger but both sides have less than 2048 pixels, the largest allowed block size is 128x128 pixels. This is the first difference of ADCTC with respect to our earlier coder [8].

While performing PS optimization, such a variant of block division into two new ones is taken that has smaller value of a cost function *E*. Ideally, an entropy would be the best cost function to be used here, but due to the fact that often we have to deal with a small block sizes (thus, not enough statistics in each block to calculate an entropy) a more robust empirically found cost function is used:

$$E = \sum_{i=1}^{N} \ln(1 + |X_i| / QS), \qquad (1)$$

where *QS* denotes a quantization step for lossy compression, *N* is a total number of DCT coefficients in blocks that relate to a given variant of division, X_i is an *i*-th among these coefficients.

The use of the cost function (1) is to considerably reduce a PS optimization time in comparison to [8]. Note that [8] requires a preliminary compression and decompression of an image for all allowable sizes of blocks in order to estimate a cost function.

Next we describe how ADCTC carries out coding NOSB. DCT coefficient NOSB is determined by the length of its binary code representation (see Fig. 2). Just this number is coded and all smaller-order bits are passed to output bit-stream without compression. Note that such a coding procedure is much faster than separate bit-plane coding [8] but it requires more delicate context modeling.

Absolute value	Binary representation	NOSB
0	0000000000000000	0
1	$000000000000000\underbrace{1}$	1
5	$0000000000000\underbrace{101}$	3
12	$000000000000\underbrace{1100}$	4
81	$000000000\underbrace{1010001}$	7

Fig. 2. Explanation of NOSB determination

Using DCT coefficients already coded till the moment, one model is selected from the set of probability models. This model is used for coding NOSB, after this updating of this model is performed. Similarly to [8], the values of coefficients that are displaced from a given one by 1, 2, or 3 positions are taken into account in the model selection (see Fig. 3).

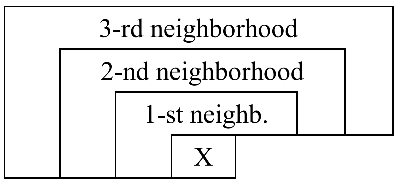

Fig. 3. Context taken into account for coding X.

A maximal NOSB is calculated in the 1-st neighborhood (maxr1), then, a maximal NOSB is found for the 2-nd (maxr2) and the 3-rd (maxr3) neighborhoods. After this, a number of probability model in the model set is determined as maxr1+*K***C*25. Here *C*25 is the maximal allowed binary number (for ADCTC *C*25:=25). The context number *K* is determined according to the following rules:

```
IF maxr2=maxr1 THEN
    IF maxr3≤maxr1 THEN K:=1
    ELSE K:=2
ELSE
    IF maxr2<maxr1 THEN K:=3
    ELSE
        IF maxr2=maxr1+1 THEN
            IF km=1 THEN K:=4
            ELSE K:=5
        ELSE K:=6.
```

Here km is the number of coefficients in the 2-nd neighborhood that have NOSB equal to maxr1+1.

Simultaneously with determination of the context *K*, additional quantization is carried out. In the case when maxr1=0 and some coded coefficient absolute value before quantization has been smaller than a threshold *Tr* (empirically for ADCTC we have set *Tr*=0.63QS) and K ≠6, then a coded coefficient is assigned zero value.

Table 1. Comparative analysis of coding efficiency

bpp	Image	JPEG	JPEG 2000	SPIHT	X-W 1999	GLBT 16x32	E-CEB	AGU 32x32	AGU MHV	ADCTC
1	Lenna	39.44	40.33	40.46	**41.21**	40.43	40.43	40.50	40.85	40.89
	Barbara	37.19	38.07	37.45	39.12	38.43	38.38	39.26	39.91	**39.96**
	Baboon	28.63	29.11	29.17	-	-	-	29.70	**30.27**	29.98
	Goldhill	36.18	36.54	36.55	37.34	36.78	37.04	37.03	**37.38**	37.31
	Peppers	37.33	38.17	38.37	-	-	-	38.33	38.91	**38.92**
	Patterns	31.01	35.80	28.83	-	-	-	38.55	43.81	**44.17**
0.5	Lenna	36.29	37.27	37.25	37.87	37.33	37.46	37.51	37.86	**37.90**
	Barbara	32.23	32.87	32.10	34.48	33.94	33.75	34.65	35.28	**35.38**
	Baboon	25.26	25.57	25.64	-	-	-	26.12	**26.39**	26.26
	Goldhill	32.88	33.24	33.13	33.75	33.42	33.60	33.65	33.81	**33.87**
	Peppers	34.82	35.80	35.82	-	-	-	35.55	36.04	**36.17**
	Patterns	22.23	28.46	20.51	-	-	-	30.66	32.51	**33.45**
0.25	Lenna	33.06	34.15	34.14	34.76	34.27	34.43	34.50	34.75	**34.95**
	Barbara	28.19	28.89	28.13	30.60	30.18	29.76	30.77	**31.21**	31.19
	Baboon	22.91	23.18	23.26	-	-	-	23.69	**23.77**	23.71
	Goldhill	30.21	30.53	30.56	30.98	30.84	30.94	31.09	31.22	**31.24**
	Peppers	32.15	33.54	33.51	-	-	-	33.32	33.95	**33.98**
	Patterns	17.36	22.60	16.25	-	-	-	25.13	26.39	**26.80**

This is an adaptive analog of a “dead zone quantization” that results in special increasing of zeroes’ number for those probability models for which this is beneficial.

For DCT coefficients with indices [0,0], [1,0], and [0,.] (all first row of DCT coefficient block), separate sets of frequency models are used. Similarly, separate sets are used for blocks that have size up to 32x32 pixels and other ones.

For coding of signs of non-zero coefficients, context modeling is used as well (see Fig. 4).

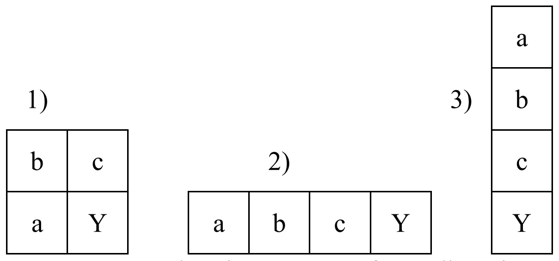

Fig. 4. Contexts taken into account for coding sign of a non-zero coefficient Y

Context 1) is taken into account as follows:

```
IF (a=0)or(b=0)or(c=0) THEN $k_1$:=2
ELSE
    IF num(+,a,b,c) IN [1,3] THEN $k_1$:=0
    ELSE $k_1$:=1;
```

The function num(+,a,b,c) calculates the number of positive values among a,b,c.

Context 2) is taken into consideration as:

```
IF (a>0)and(b>0)and(c>0) THEN $k_2$:=0
ELSE IF (a<0)and(b>0)and(c<0) THEN $k_2$:=0
ELSE IF (a<0)and(b<0)and(c<0) THEN $k_2$:=1
ELSE IF (a>0)and(b<0)and(c>0) THEN $k_2$:=1
ELSE $k_2$:=2.
```

Context 3) is taken into account similarly to Context 2), in this case the value k_3 is calculated.

Finally, the number of probability model for sign coding is calculated as $9k_1 + 3k_2 + k_3$.

3. COMPARATIVE ANALYSIS OF R-D PERFORMANCE OF THE PROPOSED CODER

In this section we compare R-D performance of the proposed ADCTC coder to other known and most advanced coders. Such comparison occurs to be complex because not all coders described in literature have available executable files. Besides, performance of some coders has not been analyzed for some test images.

First of all, we consider well established coders such as JPEG [4], JPEG2000 [15], and SPIHT [5]. For correctness of comparison, a version of JPEG that uses uniform quantization and fast post-filtering like one applied in [8] is used. Since we apply such JPEG variant, it produces larger PSNR than the conventional JPEG.

Besides, one of the best wavelet based coder [16] (X-W 1999), a coder based on lapped orthogonal transform [2] (GLBT), a coder of the same authors that operates with 8x8 blocks [9] (E-CEB) as well as our earlier coders AGU [13] and AGU-MHV [8] will be also used for comparison. The results of such a comparison for bpp=1, 0.5, and 0.25 are given in Table 1.

ADCTC and AGU-MHV in all considered cases except one provide better performance than other coders. Note that the coder [16] is not available and the results are given as in [16]. The proposed coder ADCTC produces the results that are either better or slightly worse than AGU-MHV. This evidences in favor of high quality of the designed context models avoiding a usage of slow bit-plane coding.

It is also worth noting that for JPEG and JPEG2000 decompressed image quality differs only a little (by 0.5…1 dB on the average), although this is mainly due to a post-filtering (deblocking) of JPEG decompressed images.

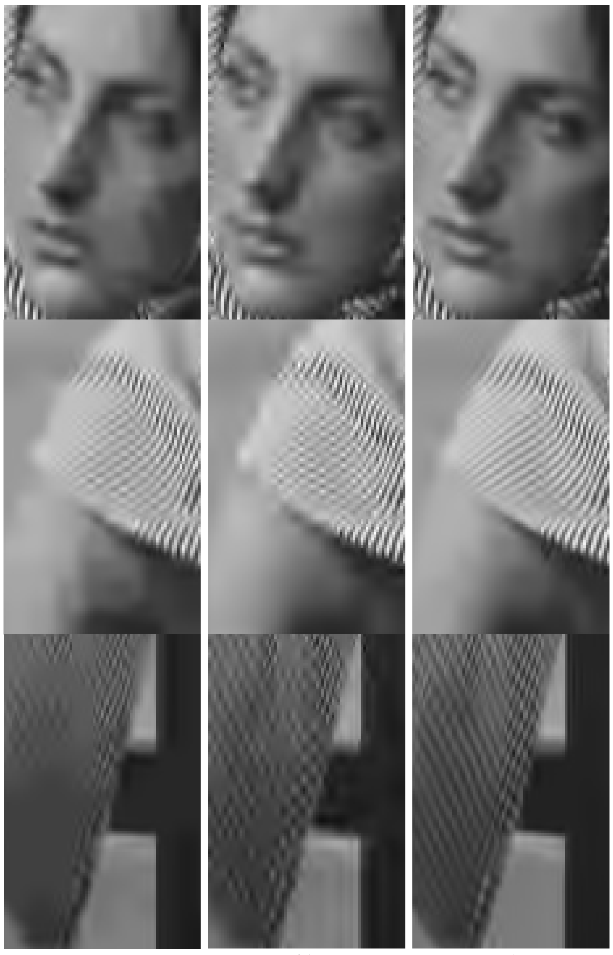
a) b) c)

Fig.5. Fragments of the decoded test image Barbara, bpp=0.25: a) JPEG, b) JPEG2000, c) ADCTC

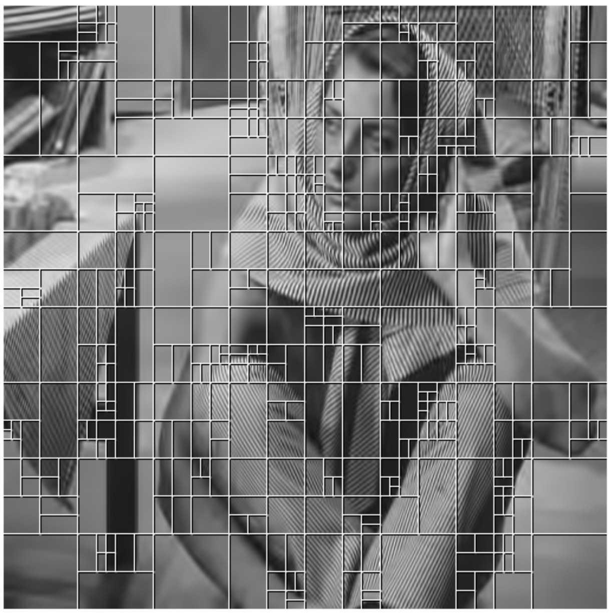
Fig.6. MHV PS for ADCTC for image Barbara bpp=0.25

Fig. 5 presents decompressed fragments for the test image Barbara for the coders JPEG, JPEG2000, and ADCT. The advantages of ADCTC especially for textured fragments can be seen well.

Fig. 6 presents the PS obtained for ADCTC for the image Barbara, bpp=1. Fig. 7 shows the PS that has been obtained for another compression ratio - bpp=0.25. As it can be seen, large size blocks for ADCTC correspond to either image homogeneous regions or to fragments that have a regular texture.

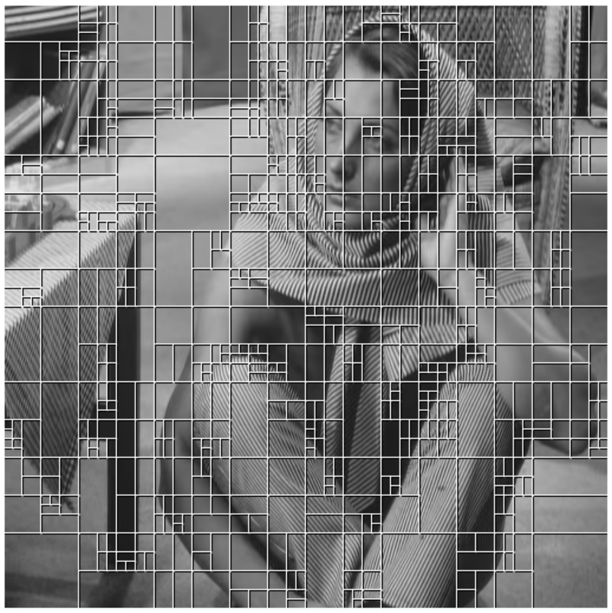
Fig.7. MHV PS for ADCTC for image Barbara, bpp=1

One more tendency is observed. If QS increases, average block size for PS also become larger.

4. EXTENDED COMPARISON OF JPEG, JPEG2000 AND ADCTC FOR LOW-RES IMAGES

In this section we will compare performance of ADCTC to JPEG and JPEG2000 on a set of 16 test images of size 512x512 pixels (available in [12]). Table 2 presents corresponding PSNR results for five different bitrates. In all cases ADCTC produces better results than JPEG2000. Due to good spatial adaptivity, ADCTC provides by more than 5 dB higher PSNR than JPEG2000.

Fig. 8 depicts the plots of PSNR vs bitrate for the compared techniques that have been obtained for entire test set. As can be seen, the ADCTC overcomes JPEG2000 by more than JPEG2000 overcomes JPEG.

In turn, Fig. 9 demonstrates that for almost entire range of considered bpp values (calculated for ADCTC), this method produces by more than 1.3 times better compression ratio than JPEG2000.

Table 2. Comparative analysis of coding efficiency of JPEG, JPEG2000 and ADCTC for set of 512x512 pixels images

Image	JPEG					JPEG2000					ADCTC				
	bpp=2	1	0.5	0.25	0.125	bpp=2	1	0.5	0.25	0.125	bpp=2	1	0.5	0.25	0.125
Airfield	35.23	30.81	27.95	25.33	23.12	36.24	31.52	28.33	25.59	23.52	**36.97**	**31.78**	**28.68**	**26.13**	**23.91**
Baboon	33.96	28.63	25.26	22.91	21.29	34.77	29.11	25.57	23.18	21.68	**35.89**	**29.98**	**26.26**	**23.71**	**21.99**
Barbara	42.41	37.19	32.23	28.19	25.17	43.79	38.07	32.87	28.89	25.76	**45.59**	**39.96**	**35.38**	**31.19**	**27.85**
Boat	40.95	36.05	32.84	29.63	26.82	41.91	36.68	33.34	30.16	27.41	**43.11**	**37.48**	**33.98**	**30.86**	**28.06**
Cartoon	45.93	35.11	28.02	23.18	20.12	49.01	37.12	29.44	24.01	20.55	**50.24**	**37.98**	**30.32**	**25.04**	**21.31**
Depart	43.48	38.14	33.93	30.55	27.60	44.61	38.94	34.43	30.93	28.06	**45.65**	**39.64**	**35.23**	**31.72**	**28.79**
Family	41.49	35.57	31.31	28.18	25.74	42.92	36.27	31.67	28.49	26.22	**43.64**	**37.01**	**32.34**	**29.00**	**26.56**
Fly	47.89	43.01	37.58	33.10	29.61	48.84	44.17	38.84	34.08	30.63	**50.11**	**44.89**	**39.54**	**34.89**	**31.27**
Goldhill	40.82	36.18	32.88	30.21	28.03	41.82	36.54	33.24	30.53	28.49	**42.89**	**37.31**	**33.87**	**31.24**	**29.08**
Grass	26.65	21.49	18.61	16.80	15.71	26.79	21.39	18.79	16.96	15.85	**28.26**	**22.46**	**19.18**	**17.20**	**15.91**
Lena	43.73	39.44	36.29	33.06	29.73	44.66	40.33	37.27	34.15	31.02	**46.00**	**40.89**	**37.90**	**34.95**	**31.91**
Map	27.11	22.09	19.30	17.53	16.49	27.15	21.80	19.26	17.65	16.65	**28.29**	**22.65**	**19.61**	**17.86**	**16.74**
Patterns	45.20	31.01	22.23	17.36	15.36	41.25	35.80	28.46	22.60	18.67	**60.10**	**44.17**	**33.45**	**26.80**	**22.30**
Peppers	41.75	37.47	35.03	32.65	29.69	42.98	38.17	35.80	33.54	30.79	**44.18**	**38.92**	**36.17**	**33.98**	**31.46**
Pole	49.91	43.83	37.21	32.06	27.93	49.83	45.53	38.37	32.98	28.84	**51.35**	**46.40**	**39.91**	**34.46**	**30.39**
Ponom	31.69	26.35	23.24	21.25	19.92	32.09	26.41	23.24	21.46	20.17	**33.23**	**27.20**	**23.75**	**21.65**	**20.34**

Table 3. Comparison of coding efficiency for JPEG, JPEG2000 and ADCTC for the set of 2288x1712 pixels images

Image	JPEG					JPEG2000					ADCTC				
	bpp=2	1	0.5	0.25	0.125	bpp=2	1	0.5	0.25	0.125	bpp=2	1	0.5	0.25	0.125
Pole	49.33	44.62	41.90	39.19	35.17	49.29	45.22	42.46	40.05	36.54	**51.17**	**46.23**	**42.95**	**40.88**	**38.33**
Stones	35.59	29.80	25.51	22.26	19.87	36.69	30.25	25.80	22.50	20.18	**38.23**	**31.75**	**26.85**	**23.01**	**20.32**
Sasha	35.65	29.96	25.72	22.58	20.25	36.73	30.22	26.08	22.97	20.64	**39.06**	**32.44**	**27.54**	**23.81**	**21.27**
Car	38.19	32.48	28.38	25.44	23.20	38.91	32.82	28.61	25.55	23.55	**40.78**	**34.39**	**29.66**	**26.12**	**23.77**
Pond	43.33	35.66	30.56	26.63	23.56	45.38	36.70	31.03	27.08	24.38	**47.23**	**38.42**	**32.80**	**28.48**	**25.17**
Stubs	42.38	36.72	32.82	29.60	27.06	43.41	37.36	33.18	30.11	27.54	**44.95**	**38.72**	**34.16**	**30.58**	**27.92**
Flowers	40.65	35.45	31.26	27.45	24.21	42.12	36.32	31.89	28.07	24.86	**43.50**	**37.49**	**32.97**	**28.90**	**25.47**
Macro	49.61	46.96	43.92	41.90	40.13	50.55	46.79	44.39	42.78	41.69	**52.90**	**48.07**	**44.90**	**43.04**	**41.94**

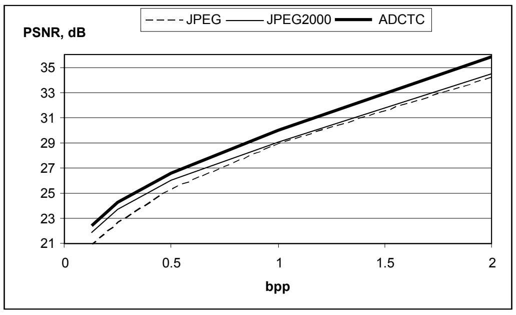

Fig.8. Comparison of the coders JPEG, JPEG2000, and ADCTC for entire set of test images

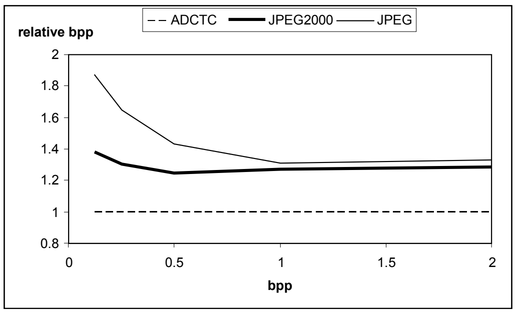

Fig.9. The plots showing how many times larger memory is required for JPEG2000 and JPEG to provide the same PSNR as for the ADCTC

5. EXTENDED COMPARISON OF JPEG, JPEG2000 AND ADCTC FOR HIGH RESOLUTION IMAGES

In the previous Section we have compared ADCTC and JPEG2000 for images of size 512x512. However, in practice it is more interesting to analyze coder's effectiveness for high resolution images of a larger size typical for modern digital cameras.

A test set of 2288x1712 pixels images is obtained by the first author of this paper using digital camera OLYMPUS C765-UZ [12]. Images have been saved in TIFF format (without any compression). The test set contains images with various characteristics, such as large homogeneous regions, highly textured images, as well as images containing a lot of details.

The compression results, collected in Table 3 clearly demonstrate advantages of ADCTC with respect to JPEG and JPEG2000 even better than for the case of 512x512 pixel images (see Fig 10 and Fig. 11).

As seen, for bpp=2 and bpp=1, typical for digital pictures, ADCTC provides by 1.4 times better compression than JPEG2000 and by 1.55 times better than JPEG.

Table 4. Comparison of coding efficiency for JPEG, JPEG2000 and ADCTC for the test image CATS

Image	JPEG					JPEG2000					ADCTC				
	bpp=2	1	0.5	0.25	0.125	bpp=2	1	0.5	0.25	0.125	bpp=2	1	0.5	0.25	0.125
Cats	50.53	42.67	36.39	32.47	29.28	42.05	40.57	36.45	32.56	29.90	**57.08**	**46.51**	**39.92**	**35.28**	**31.86**

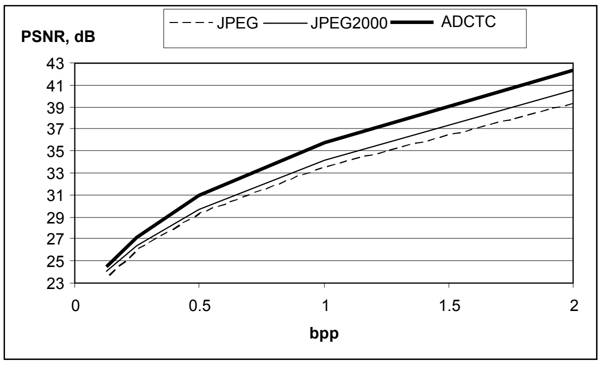

Fig.10. Comparison of the coders JPEG, JPEG2000, and ADCTC for entire set of the test images in Fig. 10

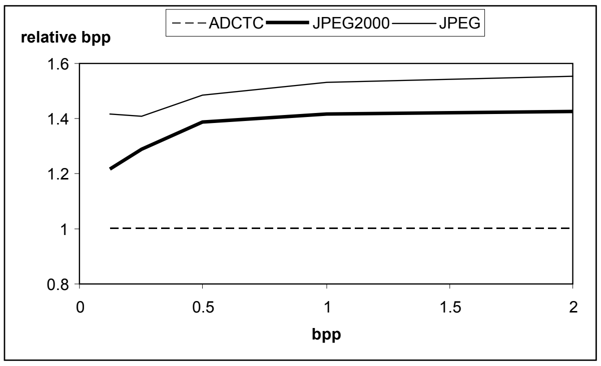

Fig.11. The plots showing how many times larger memory is required for JPEG2000 and JPEG to provide the same PSNR as for the ADCTC

On another large test image "Cats" (see Table 4) that has been used for testing a performance of JPEG2000, ADCTC produces by 6 dB better PSNR than JPEG2000 for the bit rate of 1bpp.

6. CONCLUSIONS

This paper presents effective DCT based coder ADCTC for lossy image compression. This coder has demonstrated better R-D performance than state-of-the-art coders for a wide set of test images.

ADCTC and all the sets of the test images can be downloaded from [12].

7. REFERENCES

[1] K. Rao, P. Yip, *Discrete Cosine Transform, Algorithms, Advantages, Applications*. Academic Press, 1990.

[2] T.D. Tran, T.Q. Nguyen, "*A lapped transform progressive image coder*", in IEEE Proc. of the Int. Symp. on Circuits and Syst. ISCAS '98, Vol. 4, 1998, pp. 1-4.

[3] *Wavelet Image and Video Compression*, Edited by Pankaj N. Topiwala, Boston, USA: Kluwer Acad. Publ., 1998.

[4] G.K. Wallace, *The JPEG Still Picture Compression Standard*, Comm. of the ACM, Vol. 34, 1991, pp. 30-44.

[5] A. Said, W.A. Pearlman, *A new fast and efficient image codec based on the partitioning in hierarchical trees*, IEEE Trans. on Circuits Syst. Video Technol., Vol. 6, 1996. - pp. 243-250.

[6] D. Taubman, *High performance scalable image compression with EBCOT*, IEEE Trans. Image Processing, vol. 9, 2000. - pp. 1158–1170.

[7] D. Taubman, M. Marcellin, *JPEG 2000: Image Compression Fundamentals, Standards and Practice*. Boston: Kluwer, 2002.

[8] N.N. Ponomarenko, V.V. Lukin, K.O. Egiazarian, J.T. Astola, *High Quality DCT Based Image Compression Using Partition Schemes*, IEEE Signal Processing Letters, Vol. 14, 2007, pp. 105-108.

[9] C.Tu, T.D.Tran, *Context-Based Entropy Coding of Block Transform Coefficients for Image Compression*, IEEE Trans. on Image Proc., vol. 11, 2002, pp. 1271-1283.

[10] N.N. Ponomarenko, A.V. Bazhyna, K.O. Egiazarian, *Prediction of signs of DCT coefficients in block-based lossy image compression*, Proc. of the SPIE Conf. Image Proc.: Alg. and Syst. V, SPIE Vol. 6497, January, 2007.

[11] A. Deever and S. S. Hemami, *What's your sign?: Efficient sign coding for embedded wavelet image coding*, in Proc. 2000 Data Compr. Conf., 2000, pp. 273–282.

[12] ADCTC software and set of test images: http://www.ponomarenko.info/adct.htm

[13] N.N. Ponomarenko, V.V. Lukin, K.O. Egiazarian, J.T. Astola, *DCT Based High Quality Image Compression*, in Proc. Scand. Conf. on Image Analysis, Springer Series: Lect. Notes in Comp. Sc., vol.3540, 2005, pp.1177-1185.

[14] K. Egiazarian, J. Astola, M. Helsingius, P. Kuosmanen, *Adaptive denoising and lossy compression of images in transform domain*, Journal of Electronic Imaging, vol. 8, 1999, pp. 233-245.

[15] *Kakadu JPEG 2000 SDK Home page*. [Online]. Available: http://www.kakadusoftware.com/

[16] Z. Xiong, X. Wu, *Wavelet image coding using trellis coded space-frequency quantization*, in IEEE Signal Processing Letters, Vol. 6, Issue 7, 1999, pp. 158-161.