DCT Based High Quality Image Compression

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

[1] Department 504, National Aerospace University
(Kharkov Aviation Institute),
17 Chkalova Street, 61070, Kharkov, Ukraine
lukin@xai.kharkov.ua
[2] Tampere International Center for Signal Processing,
Tampere University of Technology,
P.O.Box-553, FIN-33101, Tampere, Finland
{karen, jta}@cs.tut.fi

Abstract. DCT based image compression using blocks of size 32x32 is considered. An effective method of bit-plane coding of quantized DCT coefficients is proposed. Parameters of post-filtering for removing of blocking artifacts in decoded images are given. The efficiency of the proposed method for test images compression is analyzed. It is shown that the proposed method is able to provide the quality of decoding images higher than for JPEG2000 by up to 1.9 dB.

1 Introduction

Discrete cosine transform (DCT) [1,2] is the basis of many image compression methods. For example, the standard JPEG [3,4], for which DCT is carried out in 8x8 image blocks existed as the main image compression standard for about 10 years. However, many investigations and achieved progress in this area have dealt with applications of discrete wavelet transform (DWT). For example, the compression standard JPEG2000 [5,6] accepted quite recently is based on DWT and it commonly provides considerably better quality of decoded images than JPEG.

Aforesaid allows supposing that DWT is more appropriate transform for applying in image compression than DCT. In this paper we try to show that this is not true. Due to rather simple improvements of the base method (used in JPEG) it is possible to obtain decoded images quality better than for JPEG2000.

There are three basic modifications introduced by us compared to JPEG. First, an image is divided into 32x32 pixel blocks instead of 8x8 for conventional JPEG. Second, the quantized DCT coefficients are divided into bit-planes; the bit values are coded according to complex probability models that take into account the presence of correlation between values of neighbor coefficients in blocks and between the values of the corresponding coefficients of neighbor blocks. Third, DCT based filtering [7] is used as post-processing for removal of blocking artifacts from decoded images and, thus, for increasing decoded image quality.

H. Kalviainen et al. (Eds.): SCIA 2005, LNCS 3540, pp. 1177 – 1185, 2005.

2 Coding and Decoding Schemes

The block-diagram of image coding for the proposed approach is depicted in Fig. 1.

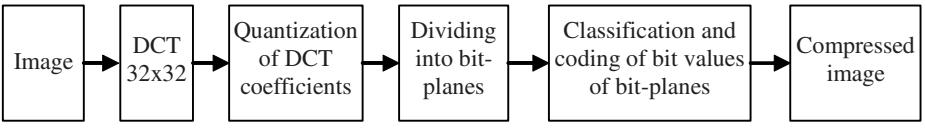

Fig. 1. The block-diagram of image coding

An image to be compressed is divided into 32x32 pixel blocks. Then, DCT for pixel values of each block is computed. After this, the quantization of DCT coefficients of image blocks is carried out. At this stage the basic losses are introduced into compressed image. Larger quantization step (QS) provides larger compression ratio (CR) and simultaneously it leads to larger losses. In this paper it is proposed to use uniform quantization that ensures the best results within the structure of the considered method.

Then, the division of quantized DCT coefficients into bit-planes is carried out. The obtained bit-planes are coded in the order starting from higher bits to lower ones. While coding each next plane, the values of bits of earlier coded planes are taken into account. A coded bit is referred to one or another group of bits according to the values of already coded bits. For each group of bits, individual probability model is used for dynamic arithmetic coding (see Section 3).

The block-diagram of image decoding is presented in Fig. 2 where IDCT denotes inverse DCT.

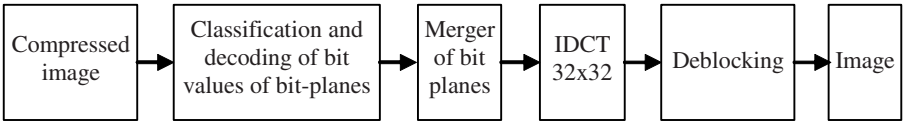

Fig. 2. The block-diagram of image decoding

As seen, at image decoding stage all steps are repeated in reverse order. Besides, at the final step the operation of decoded image filtering is added (see Section 4 for more details).

3 Bit-Plane Coding

Thus, after calculation of DCT in 32x32 blocks and quantization of obtained coefficients, we have an array of integer valued DCT coefficients. Divide the array of absolute values of DCT coefficients into n bit-planes, where n is the number of the highest bit-plane in which there are non-zero values. Coding begins with the bit-plane n and comes to an end by bit-plane 1.

The signs of non-zero DCT coefficients are practically random variables with approximately equal probabilities. Therefore, they are allocated into a separate array (one bit for each sign) and transferred to the output stream at once.

Let $P^k_{l,m}(i,j)$ defines a bit value of a bit-plane k of a coefficient with the index i,j of the block of an image with the index l, m, where k=1..n, i,j=1..32, l=1..L, m=1..M, L,M denotes the number of image blocks for vertical and horizontal directions. We introduce the following conditions which are used for classification of bits of bit-planes:

1) $C1(k,l,m,i,j)=true$, if $1 \in \{P^{k+1}_{l,m}(i,j),..., P^n_{l,m}(i,j)\}$. This condition is assigned *true* if, at least, one bit among earlier coded higher bit planes is equal to 1.

2) $C2(k,l,m,i,j)=true$, if $1 \in \{P^{k+2}_{l,m}(i,j),..., P^n_{l,m}(i,j)\}$. This condition is *true* if, without taking into account the previously coded higher bit-plane, the bit with these indices was equal to 1. If the condition *C2* is *true* then the current bit with approximately equal probability can be equal either to 0 or to 1. If the condition *C1* is *true* and the condition *C2* is *false* then the probability of zero for the current bit is considerably larger than the probability to be equal to 1.

3) $C3(k,l,m,i,j)=true$, if $1 \in \{P^k_{l,m}(i,j),..., P^n_{l,m}(i,j)\}$. This condition is *true* if in this or in, at least, one of earlier coded higher bit planes the bit with these indices was equal to 1. The condition C3 can be checked for those bits neighboring the coded bit that till the current moment have been already coded. Here and below only the values of those bits can be checked that have been already coded. This is important for providing an opportunity of decoding. At decoding stage those bits that have been coded earlier are decoded earlier as well and they can be checked in conditions.

4) $C4(k,l,m,i,j)=true$, if $P^{k+1}_{l,m}(i,j)=1$. This condition is *true* if in the previously coded bit plane the bit with these indices was equal to 1.

5) $C5(k,l,m,i,j)=true$, if $P^k_{l,m}(i,j)=1$.

6) $C6(k,l,m,i,j)=true$, if $true \in \{C1(k,l,m,i-1,j-1), C1(k,l,m,i-1,j), C1(k,l,m,i-1,j+1), C1(k,l,m,i,j-1), C1(k,l,m,i,j+1), C1(k,l,m,i+1,j-1), C1(k,l,m,i+1,j), C1(k,l,m,i+1,j+1)\}$. This condition is *true* if for, at least, one of neighboring bits there is unity in higher bit planes.

7) $C7(k,l,m,i,j)=true$, if $true \in \{C5(k,l,m,i-1,j-1), C5(k,l,m,i-1,j), C5(k,l,m,i-1,j+1), C5(k,l,m,i,j-1)\}$. This condition is *true* if, at least, one among neighboring and already coded bits of this bit-plane was equal to 1.

8) $C8(k,l,m,i,j)=true$, if $true \in \{C3(k,l,m,i-2,j-2), C3(k,l,m,i-2,j-1), C3(k,l,m,i-2,,j), C3(k,l,m,i-2,j+1), C3(k,l,m,i-1,j-2), C3(k,l,m,i-1,j+2), C3(k,l,m,i,j-2), C3(k,l,m,i,j+2), C3(k,l,m,i+1,j-2), C3(k,l,m,i+1,j+2), C3(k,l,m,i+2,j-2), C3(k,l,m,i+2,j-1), C3(k,l,m,i+2,j), C3(k,l,m,i+2,j+1), C3(k,l,m,i+2,j+2)\}$. This condition is *true* if there was, at least, one unity in this or higher bit planes for already coded bits displaced from the coded bit by 2 rows or 2 columns.

9) $C9(k,l,m,i,j)=true$, if $true \in \{C3(k,l,m,i-3,j-3), C3(k,l,m,i-3,j-2), C3(k,l,m,i-3,,j-1), C3(k,l,m,i-3,j), C3(k,l,m,i-3,j+1), C3(k,l,m,i-3,j+2), C3(k,l,m,i-3,j+3), C3(k,l,m,i-2,j-3), C3(k,l,m,i-2,j+3), C3(k,l,m,i-1,j-3), C3(k,l,m,i-1,j+3), C3(k,l,m,i,j-3), C3(k,l,m,i,j+3), C3(k,l,m,i+1,j-3), C3(k,l,m,i+1,j+3), C3(k,l,m,i+2,j-3), C3(k,l,m,i+2,j+3), C3(k,l,m,i+3,j-3), C3(k,l,m,i+3,j-2), C3(k,l,m,i+3,j-1), C3(k,l,m,i+3,j), C3(k,l,m,i+3,j+1), C3(k,l,m,i+3,j+2), C3(k,l,m,i+3,j+3)\}$. This condition is *true* if there was unity in this or higher bit planes for already coded bits displaced from the coded bit by 3 rows or 3 columns.

10) $C10(k,l,m,i,j)=true$, if $true \in \{C3(k,l-1,m-1,i,j), C3(k,l-1,m,i,j), C3(k,l-1,m+1,i,j), C3(k,l,m-1,i,j), C3(k,l,m+1,i,j), C3(k,l+1,m-1,i,j), C3(k,l+1,m,i,j), C3(k,l+1,m+1,i,j)\}$. This condition is *true* if there was unity in this or in higher bit planes for bits in neighbor blocks. This condition allows taking into consideration correlation for bits having identical indices and belonging to image neighbor blocks.

11) $C11(k,l,m,i,j)=true$, if $(C2(k,l,m,i,j)=false)$and$(C6(k+1,l,m,i,j)=false)$. The checking of this condition allows classifying more reliably the bit for which in the previously coded bit plane there was unity.

12) $C12(k,l,m,i,j)=\begin{cases}1, & C5(k,l,m,i,j)=true\\ 0, & C5(k,l,m,i,j)=false\end{cases}$.

13) $C13(k,l,m,i,j)=true$, if $1 = C12(k,l,m,i-1,j-1) + C12(k,l,m,i-1,j) + C12(k,l,m,i-1,j+1) + C12(k,l,m,i,j-1)$.

14) $C14(k,l,m,i,j)=true$, if $k=1$.

Fig. 3 presents the flowchart of bit value classification by checking the aforementioned conditions (*PM*X - probability model number X).

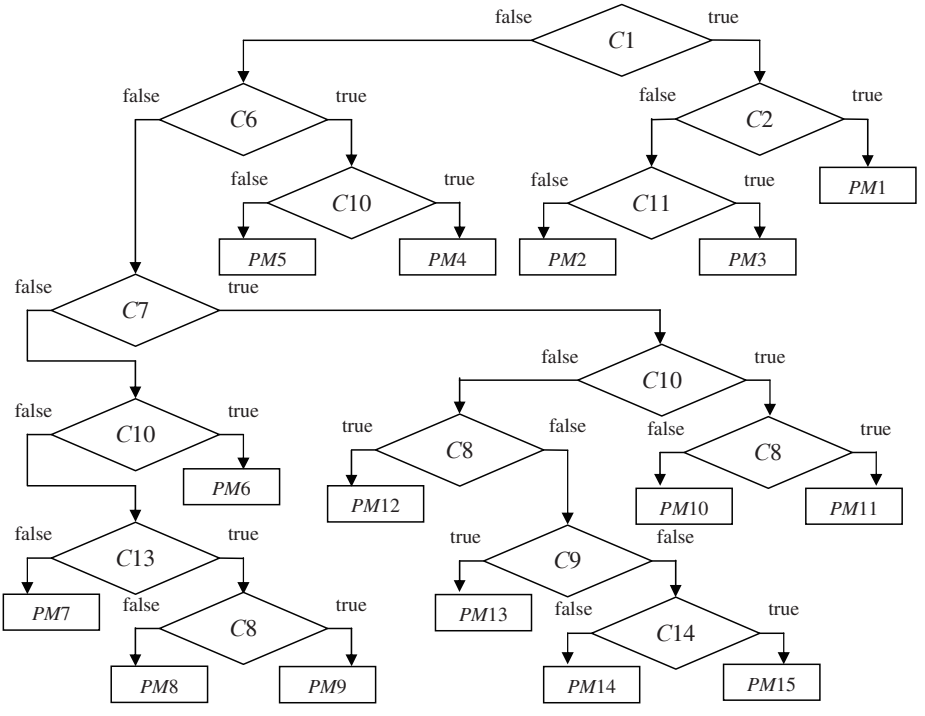

Fig. 3. Flow chart of classification of coded bits of bit-planes

Totally according to given classification a bit can be referred to one of fifteen probability models. For each model after coding the current bit the counters of 0 and 1 are corrected, and they are used for coding next bits referred to this model. For coding it

is proposed to use the dynamic version of arithmetic coding [8,9] that is the most effective for the considered case.

Let us consider the obtained classification more in detail. To *PM*1 and *PM*2 those bits are referred that for higher bit planes had unities and for which the probabilities of being equal to 0 and 1 are practically equal. To *PM*3 those bits are referred for which there was unity in previously coded plane and the probability of unity was low. Because of this, for *PM*3 the probability of 0 is larger than being equal to 1.

To the model *PM*4 those bits are referred that have no unities in higher bit-planes but there are neighbor bits with unities in higher bit planes and there are unities in the corresponding bits of image neighbor blocks. For *PM*4 the probabilities of 1 and 0 are rather close, and the bits referred to this model are compressed poorly. The difference between the models *PM*4 and *PM*5 consists in the following. To *PM5* those bits are referred that have no unities in the corresponding bits of image neighbor blocks. For the model *PM5* there are considerably more zeros than unities, and the corresponding bits are compressed considerably better.

The bits of the models *PM*6-*PM*9 differ from the bits of the models *PM*4 and *PM*5. For the former ones there are no neighbor bits with unities in higher bit planes, but there are unities for neighbor bits in the current coded plane. For the models *PM*6-*PM*9 the probability of unities is considerably smaller than for the models *PM*4 and *PM*5. Because of this, these data are compressed better. Those bits are referred to the model *PM*6 that have unities in the corresponding bits of image neighbor blocks. The bits of this model are compressed in the worst way among the bits that belong to the group of the models *PM*6-*PM*9. For the models *PM*7-*PM*9 there are no unities in the corresponding bits of image neighbor blocks. For the bits referred to the model *PM*7 the number of unities in the neighbor bits is larger than 1. For the models *PM*8 and *PM*9 there is only one unity in the neighbor bits. Because of this, the probability of unities for them is even smaller than for the model PM7. Division of bits between the models PM8 and PM9 is accomplished using the condition *C*8 that allows taking into account the presence of unities in the bits displaced from the coded one by 2 rows and 2 columns. Due to this, the bits of the model *PM*8 for which *C*8=*false* are compressed best of all among the bits of the models *PM*6-*PM*9.

The bits of the models *PM*10-*PM*15 differ from the bits of the models *PM*4-*PM*9 by the following. For the former ones there are no unities either in higher bit planes or in the current coded plane. The bits of the models *PM*10-*PM*15 are compressed very well, however, additional division of such bits into several models lead to considerable increasing of CR. Those bits are referred to the models *PM*10, *PM*11 that have unity in the corresponding bits of image neighbor blocks. The bits of the model *PM*10 are compressed slightly better since for them there are no unities in the bits displaced from the coded one by 2 rows and 2 columns. For the model *PM*13, there are unities only in bits displaced from the coded one by 3 rows and 3 columns. For the models *PM*14 and *PM*15 there are no unities in the checked area. Such bits are compressed in the best way (most efficiently). The difference between these models consists in the following. To the model *PM*15 those bits are referred that belong to the lowest bit-plane (k=1). We propose to avoid coding the bits of the model *PM*15 (they all are

considered equal to 0). This is analogous to «dead zone» in quantization. But in our case, this occurs to be effective due to selectivity of its application.

Before starting coding each bit plane, the counters of unities and zeros for the models *PM*1-*PM*14 are initialized as unities, i.e. the models of each coded plane are independent. Different copies of the models *PM*1-*PM*14 are used for different regions of image blocks. For the bits of DCT coefficient with the indices i=1, j=1 (this is the quantized value of the block mean) a separate copy of the models *PM*1-*PM*14 is used. The statistical characteristics of this DCT coefficient considerably differ from statistics of other DCT coefficients. A separate copy of the models *PM*1-*PM*14 is also used for the first (upper) row of block DCT coefficients. This is explained by the fact that for these coefficients there is only one earlier coded bit. This leads to considerable difference of bit distribution between the models. For all other DCT coefficients of a block (and they are the basic amount of data) the third copy of the models *PM*1-*PM*14 is used.

The proposed classification is obtained by experimental studies of efficiency of various ways to divide bits into classes for different test images and QS. Probably, more effective variant of such classification can be found. In practice, simpler variants of classification can be used in order to increase coding speed. For example, the absence of checking the condition *C10* (in this case one does not take into account the correlation between neighbor blocks of an image) results in increasing the size of compressed image by 1-3 %.

If one does not check the condition *C*9 (this condition deals with correlation of bits displaced from the coded one by 3 rows and 3 columns) the size of coded image increases by 1-1.5%. If one also does not check the condition *C*8 (this condition deals with correlation of bits displaced from the coded one by 2 rows and 2 columns), this leads to the increasing of coded image size by 3-7%.

Let us mention one important point once again. For the used variant that includes the *PM15*, the losses of image quality occur not only at DCT coefficient quantization step, but also (though in much smaller degree), at the step of bit values coding for bit-planes. If one avoids using the model *PM15* this does not lead to any additional losses at this step.

4 Filtering for Removal of Blocking Artifact

For blocking effect reduction in decoded images, we employ an approach described in [7]. This approach presumes the use of DCT based filter for additive noise removal [10]. In the considered case, the noise to be removed is the quantization noise. The size of a sliding window of the DCT based filter is 8x8. For each position of the sliding window, DCT is carried out, then DCT coefficients having absolute values smaller than preset threshold are assigned zero values (hard thresholding). After this, inverse DCT is executed.

Spatially invariant denoising is employed. One problem is the setting of the threshold. For our application we recommend to set the threshold equal to QS/2 (note that we know QS a priori).

The use of post-filtering in our case allows increasing quality of the decoded images by 0.5-1 dB. The decoding time increases by 30-40 %.

5 Numerical Simulations

The quality of compression for the proposed method was analyzed for 512x512 grayscale images in comparison to JPEG2000 (Kakadu coder by D.Taubman [6] has been employed). The practical realization of our method in programming language Delphi (the coder has the name AGU) is accessible to downloading from the address http://www.cs.tut.fi/~karen/agucoder.htm. This version is intended for coding only 512x512 grayscale images in RAW format (without heading). The used set of test images is accessible to downloading from the same address.

The quality of decoded images was compared for CRs equal 8, 16, 32 and 64. As quality criterion, the peak signal to noise ratio was used: $PSNR = 10\lg(255^2 / [\sum_{i=1}^{I}\sum_{j=1}^{J}(I_{ij} - I_{ij}^e)^2 / IJ])$, where I, J denote the image size, I_{ij}^e is the value of the ij-th pixel of original image, and I_{ij} defines the ij-th pixel value for the analyzed (decompressed) image. Table 1 presents the obtained *PSNRs* for the considered methods.

Table 1. The quality of the test image compression for JPEG2000 and AGU, *PSNR*, dB

Image	CR=8		CR=16		CR=32		CR=64	
	JPEG2000	AGU	JPEG2000	AGU	JPEG2000	AGU	JPEG2000	AGU
Lenna	40.33	40.52	37.27	37.46	34.15	34.51	31.02	31.50
Barbara	38.07	39.26	32.87	34.65	28.89	30.77	25.87	27.55
Baboon	29.11	29.70	25.57	26.12	23.18	23.69	21.68	22.01
Goldhill	36.54	37.03	33.24	33.65	30.53	31.09	28.49	28.97
Peppers	38.17	38.33	35.80	35.55	33.54	33.32	30.79	30.90

As seen from data presented in Table 1, in overwhelming majority of the considered situations AGU outperforms JPEG2000 by quality of the decoded images. The only exceptions are CR=16 and CR=32 for the image Peppers for which JPEG2000 provides PSNRs that are better than for AGU by 0.2-0.25 dB. At the same time, for more complex images like Baboon and Goldhill the benefit of AGU for all CRs is 0.3-0.6 dB. And for the image Barbara that differs from other images by the presence of a large number of textural regions the advantage of AGU is 1.2-1.9 dB.

Image compression performance can be also compared for identical quality of decompressed images. For example, for *PSNR=28.89* dB JPEG2000 compresses the image Barbara by 32 times while AGU compresses this image by 46.7 times, that is by 1.46 times better. For *PSNR=25.87* dB AGU compresses this image by 101.5 times, that is 1.59 times better than JPEG2000 for which CR=64.

The presented data confirm that the proposed method outperforms JPEG2000 in image coding quality. The smoother is the image, the less difference of coding quality is observed for JPEG2000 and AGU. And the more complex and textural is the image, the difference of coding quality is larger.

The fragment of decoded image Barbara for JPEG2000 and AGU is shown in Fig. 4.

a)

b)

Fig. 4. A fragment of the decoded image Barbara, CR=32 a) JPEG2000, *PSNR*=28.89 dB b) AGU, *PSNR*=30.77 dB

6 Conclusions

The carried out studies show that the method proposed and described in this paper provides better quality of decoded images than JPEG2000 in most of practical situations. And its superiority for complex textured images in some cases can reach 1.9 dB.

The proposed method is obtained by rather simple modifications of JPEG, in which DCT serves as its core. This indicates that DCT is at least not worse transformation for use in image compression than DWT used as the basis of JPEG2000.

For software realizations of AGU (not optimized), the required computation time is by about 15-20 times larger than for standard JPEG. The ways to speed up AGU can be studied in future. In particular, algorithms of fast integer valued approximation of DCT in 32x32 blocks seem to lead to considerable decreasing of computation time.

In future it is possible to consider the use of partition schemes [11] that make image compression methods more adaptive. Besides, a perspective direction is the use of DCT based image compression methods directed on reduction of blocking effect such as lapped orthogonal transforms [12, 13].

References

1. Ahmed, N., Natarajan T., Rao K. R.: Discrete cosine transform. In: IEEE Transactions on Computers, Vol. 23, (1974) 90-93
2. Rao, K., Yip P.: Discrete Cosine Transform, Algorithms, Advantages, Applications. In: Academic Press (1990)
3. Wallace, G. K.: The JPEG Still Picture Compression Standard. In: Comm. Of the ACM, Vol. 34, No.4 (1991)
4. Pennebaker, W. B., Mitchell, J. L.: JPEG Still Image Data Compression Standard. In: Van Nostrand Reinhold, New York (1993)
5. Christopoulos, C., Skodras, A., Ebrahimi, T.: The JPEG2000 still image coding system: an overview. In: IEEE Trans. on Consumer Electronics, Vol. 46, Issue: 4 (2000) 1103-1127
6. Taubman, D., Marcellin, M.: JPEG 2000: Image Compression Fundamentals, Standards and Practice. In: Boston: Kluwer (2002)
7. Egiazarian, K., Helsingius, M., Kuosmanen, P., Astola, J.: Removal of blocking and ringing artifacts using transform domain denoising. In: Proc. of ISCAS'99, Vol. 4, (1999) 139-142
8. Rissänen, J.: Generalized kraft inequality and arithmetic coding. In: IBM J. Res. Develop., Vol. 20, (1976) 198-203
9. Langdon, G.G., Rissanen, J.J.: A simple general binary source code. In: IEEE Trans. Inf. Theory, IT-28 (1982) 800-803
10. Yaroslavsky, L.: Local Adaptive Filtering in Transform Domain for Image Restoration, Enhancement and Target Location. In: 6th Int. Workshop on Digital Image Processing and Computer Graphics (DIP-97), SPIE volume 3346, (1997) 2-17
11. Ponomarenko N., Lukin V., Egiazarian K., Astola J.: Partition Schemes in DCT Based Image Compression. In: Technical Report 3-2002, ISBN 952-15-0811-6, Tampere University of Technology, Finland, (2002)
12. Malvar, H.S., Staelin, D.H.: The LOT: transform coding without blocking effects. In: IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 37, (1989) 553-559
13. Tran, T.D., de Queiroz, R., Nguyen, T.Q.: The generalized lapped biorthogonal transform. In: Proceedings of the ICASSP'98, Vol.3, (1998) 1441-1444