The JPEG Standard

Jiun-De Huang
E-mail: r95942104@ntu.edu.tw
Graduate Institute of Communication Engineering
National Taiwan University, Taipei, Taiwan, ROC

Abstract

JPEG (Joint Photographic Experts Group) is an international compression standard for continuous-tone still image, both grayscale and color. This standard is designed to support a wide variety of applications for continuous-tone images. Because of the distinct requirement for each of the applications, the JPEG standard have two basic compression methods. The DCT-based mathod is specified for lossy compression, and the predictive mathod is specified for lossless compression. A simple lossy technique called baseline, which is a DCT-based methods, has been widely used today and is sufficient for a large number of applications. In this paper, we will simply introduce the JPEG standard and focuses on the baseline method.

1 Introduction

The JPEG standard is a collaboration among the International Telecommunication Union (ITU), International Organization for Standardization (ISO), and International Electrotechnical Commission (IEC). Its official name is "ISO/IEC 10918-1 Digital compression and coding of continuous-tone still image", and "ITU-T Recommendation T.81". JPEG have the following modes of operations :

(a) **Lossless mode:** The image is encoded to guarantee exact recovery of every pixel of original image even though the compression ratio is lower than the lossy modes.
(b) **Sequential mode:** It compresses the image in a single left-to-right, top-to-bottom scan.
(c) **Progressive mode:** It compresses the image in multiple scans. When transmission time is long, the image will display from indistinct to clear appearance.
(d) **Hierarchical mode:** Compress the image at multiple resolutions so that the lower resolution of the image can be accessed first without decompressing the whole resolution of the image.

The last three DCT-based modes (b, c, and d) are lossy compression because precision limitation to compute DCT and the quantization process introduce distortion in the reconstructed image. The lossless mode uses predictive method and does not have quantization process. The hierarchical mode can use DCT-based coding or pridictive coding optionally. The most widely used mode in practice is is called the baseline JPEG system, which is based on sequential mode, DCT-based coding and Huffman coding for entropy encoding. Fig. 1 is the block diagram of baseline system.

The JPEG standard defines only the syntax of the compressed bitstream. It does not specify any thing about file format. Another standard called JFIF (JPEG File Interchange Format), created by IJG (Independend JPEG Group), make a description about how to transform a JPEG stream to a file that is suit to be saved or transmission in computer.

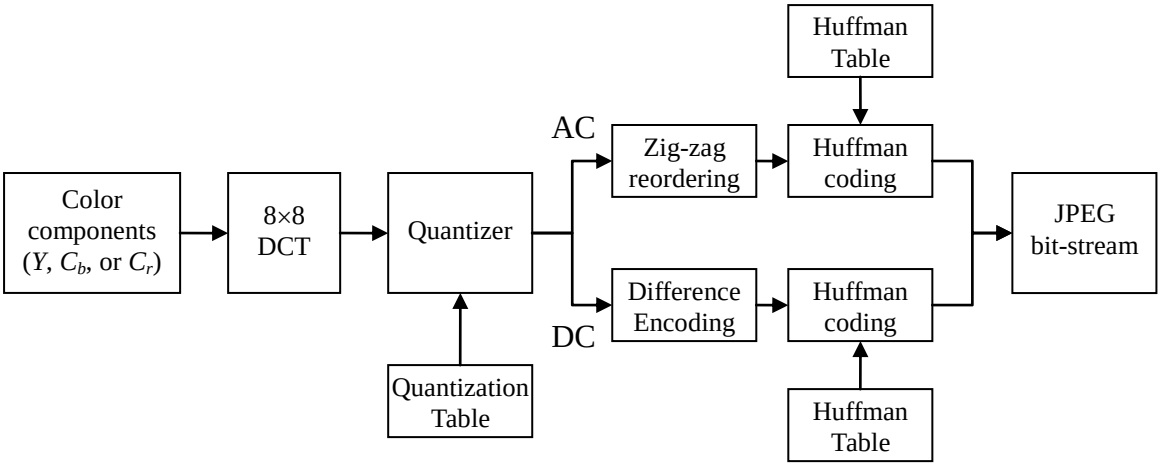


Fig. 1 Baseline JPEG encoder

2 Color Space Conversion and Downsampling

In order to achieve good compression performance, correlation between the color components is first reduced by converting the *RGB* color space into a decorrelated color space. In baseline JPEG, a *RGB* image is first transformed into a luminance-chrominancc color space such as YC_bC_r. The advantage of converting the image into luminance-chrominance color space is that the luminance and chrominance components are very much decorrelated between each other. Moreover, the chrominance channels contain much redundant information and can easily be subsampled without sacrificing any visual quality for the reconstructed image. The transformation from *RGB* to YC_bC_r, is based on the following mathematical expression:

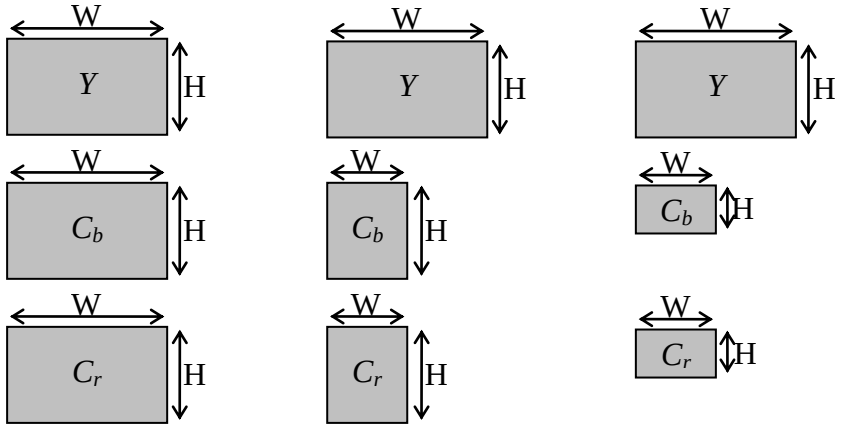
$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299000 & 0.587000 & 0.114000 \\ -0.168736 & -0.331264 & 0.500002 \\ 0.500000 & -0.418688 & -0.081312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \tag{1}$$

The value $Y = 0.299R + 0.587G + 0.114B$ is called the luminance. It is the value used by monochrome monitors to represent an *RGB* colour. Physiologically, it represents the intensity of an *RGB* color perceived by the eye. The formula is like a weighted-filter with different weights for each spectral component. The eye is most sensitive to the Green component then it follows the Red component and the last is the Blue component. The values C_b and C_r are called chromimance values and represent 2 coordinates in a system which measures the nuance and saturation of the color. These values indicate how much blue and how much red are in that color, respectively. Accordingly, the inverse transformation from YC_bC_r to *RGB* is :

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.0 & 0.0 & 1.40210 \\ 1.0 & -0.34414 & -0.71414 \\ 1.0 & 1.77180 & 0.0 \end{bmatrix} \begin{bmatrix} Y \\ C_b - 128 \\ C_r - 128 \end{bmatrix} \tag{2}$$

Because the eye seems to be more sensitive at the luminance than the chrominance, luminance is taken in every pixel while the chrominance is taken as a medium value for a 2×2 block of pixels. And this way will result a good compression ratio with almost no loss in visual perception of the new sampled image. There are three color format in the baseline system :

(a) **4:4:4 format:** The chrominance components have identical vertical and horizontal resolution as the luminance component.
(b) **4:2:2 format:** The chrominance components have the same vertical resolution as the luminance component, but the horizontal resolution is half one.
(c) **4:2:0 format:** Both vertical and horizontal resoltion of the chrominance component is half of the luminance component.



(a) 4:4:4 (b) 4:2:2 (c) 4:2:0

Fig. 2 Three color formats in the baseline system

3 Discrete Cosine Transform

To apply the DCT, the image is divided into 8×8 blocks of pixels. If the width or height of the original image is not divisible by 8, the encoder should make it divisible. The 8×8 blocks are processed from left-to-right and from top-to-bottom.

The purpose of the DCT is to transform the value of pixels to the spatial frequencies. These spatial frequencies are very related to the level of detail present in an image. High spatial frequencies corresponds to high levels of detail, while lower frequencies corresponds to lower levels of detail. The mathematical definition of DCT is :

Forward DCT :

$$F(u,v) = \frac{1}{4} C(u)C(v) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos\left[\frac{\pi(2x+1)u}{16}\right] \cos\left[\frac{\pi(2y+1)v}{16}\right]$$

$$\text{for } u = 0,...,7 \text{ and } v = 0,...,7 \qquad (3)$$

$$\text{where } C(k) = \begin{cases} 1/\sqrt{2} & \text{for } k = 0 \\ 1 & \text{otherwise} \end{cases}$$

Inverse DCT :

$$f(x,y)=\frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7}C(u)C(v)F(u,v)\cos\left[\frac{\pi(2x+1)u}{16}\right]\cos\left[\frac{\pi(2y+1)v}{16}\right] \quad (4)$$

for $x=0,...,7$ and $y=0,...,7$

The $F(u,v)$ is called the DCT coefficient, and the DCT basis is :

$$\omega_{x,y}(u,v)=\frac{C(u)C(v)}{4}\cos\left[\frac{\pi(2x+1)u}{16}\right]\cos\left[\frac{\pi(2y+1)v}{16}\right] \quad (5)$$

Then we can rewrite the inverse DCT to :

$$f(x,y)=\sum_{u=0}^{7}\sum_{v=0}^{7}F(u,v)\omega_{x,y}(u,v) \quad \text{for } x=0,...,7 \text{ and } y=0,...,7 \quad (6)$$

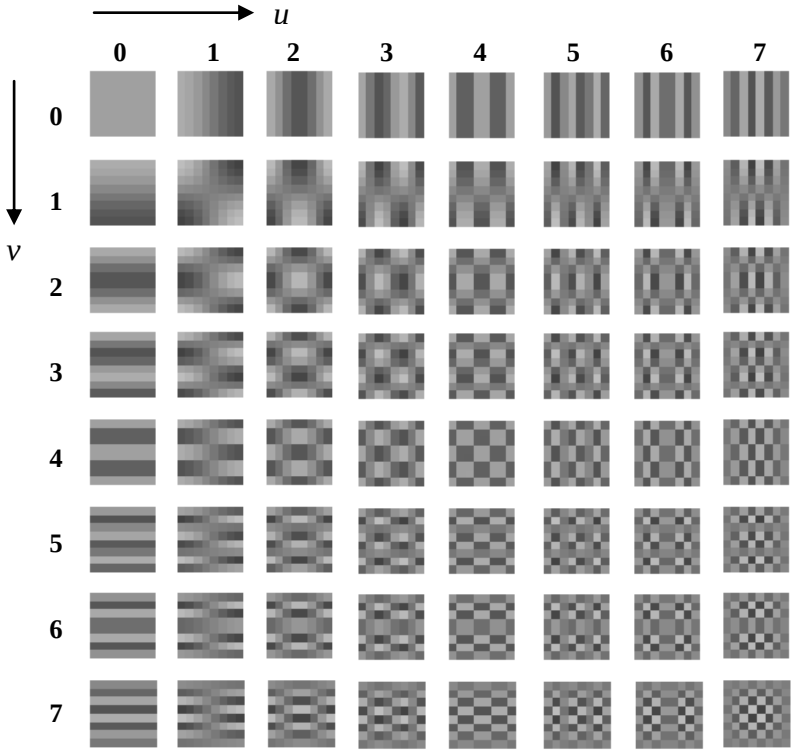


Fig. 3 The 8×8 DCT basis $\omega_{x,y}(u,v)$

48	39	40	68	60	38	50	121
149	82	79	101	113	106	27	62
58	63	77	69	124	107	74	125
80	97	74	54	59	71	91	66
18	34	33	46	64	61	32	37
149	108	80	106	116	61	73	92
211	233	159	88	107	158	161	109
212	104	40	44	71	136	113	66

699.25	43.18	55.25	72.11	24.00	-25.51	11.21	-4.14
-129.78	-71.50	-70.26	-73.35	59.43	-24.02	22.61	-2.05
85.71	30.32	61.78	44.87	14.84	17.35	15.51	-13.19
-40.81	10.17	-17.53	-55.81	30.50	-2.28	-21.00	-1.26
-157.50	-49.39	13.27	-1.78	-8.75	22.47	-8.47	-9.23
92.49	-9.03	45.72	-48.13	-58.51	-9.01	-28.54	10.38
-53.09	-62.97	-3.49	-19.62	56.09	-2.25	-3.28	11.91
-20.54	-55.90	-20.59	-18.19	-26.58	-27.07	8.47	0.31

(a) $f(x,y)$: 8×8 values of luminance (b) $F(u,v)$: 8×8 DCT coefficients

Fig. 4 An example of DCT coefficients for a 8×8 block

4 Quantization

The transformed 8×8 block now consists of 64 DCT coefficients. The first coefficient $F(0,0)$ is the DC component and the other 63 coefficients are AC component. The DC component $F(0,0)$ is essentially the sum of the 64 pixels in the input 8×8 pixel block multiplied by the scaling factor $(1/4)C(0)C(0)=1/8$ as shown in equation 3 for $F(u,v)$.

The next step in the compression process is to quantize the transformed coefficients. Each of the 64 DCT coefficients are uniformly quantized. The 64 quantization step-size parameters for uniform quantization of the 64 DCT coefficients form an 8×8 quantization matrix. Each element in the quantization matrix is an integer between 1 and 255. Each DCT coefficient $F(u,v)$ is divided by the corresponding quantizer step-size parameter $Q(u,v)$ in the quantization matrix and rounded to the nearest integer as :

$$F_q(u,v) = Round\left(\frac{F(u,v)}{Q(u,v)}\right) \tag{7}$$

The JPEG standard does not define any fixed quantization matrix. It is the prerogative of the user to select a quantization matrix. There are two quantization matrices provided in Annex K of the JPEG standard for reference, but not requirement. These two quantization matrices are shown below :

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

(a) Luminance quantization matrix

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

(b) Chrominance quantization matrix

Fig. 5 Quantization matrix

699.25	43.18	55.25	72.11	24.00	-25.51	11.21	-4.14
-129.78	-71.50	-70.26	-73.35	59.43	-24.02	22.61	-2.05
85.71	30.32	61.78	44.87	14.84	17.35	15.51	-13.19
-40.81	10.17	-17.53	-55.81	30.50	-2.28	-21.00	-1.26
-157.50	-49.39	13.27	-1.78	-8.75	22.47	-8.47	-9.23
92.49	-9.03	45.72	-48.13	-58.51	-9.01	-28.54	10.38
-53.09	-62.97	-3.49	-19.62	56.09	-2.25	-3.28	11.91
-20.54	-55.90	-20.59	-18.19	-26.58	-27.07	8.47	0.31

(a) $F(u,v)$: 8×8 DCT coefficients

44	4	6	5	1	-1	0	0
-11	-6	-5	-4	2	0	0	0
6	2	4	2	0	0	0	0
-3	1	-1	-2	1	0	0	0
-9	-2	0	0	0	0	0	0
4	0	1	-1	-1	0	0	0
-1	-1	0	0	1	0	0	0
0	-1	0	0	0	0	0	0

(b) $F_q(u,v)$: After quantization

Fig. 6 An example of quantization for a 8×8 DCT coefficients

The quantization process has the key role in the JPEG compression. It is the process which removes the high frequencies present in the original image. We do this because of the fact that the eye is much more sensitive to lower spatial frequencies than to higher frequencies. This is done by dividing values at high indexes in the vector (the amplitudes of higher frequencies) with larger values than the values by which are divided the amplitudes of lower frequencies. The bigger values in the quantization table is the bigger error introduced by this lossy process, and the smaller visual quality.

Another important fact is that in most images the color varies slow from one pixel to another. So, most images will have a small quantity of high detail to a small amount of high spatial frequencies, and have a lot of image information contained in the low spatial frequencies.

5 Zig-Zag Reordering

After doing the DCT transform and quantization over a block of 8x8 values, we have a new 8×8 block. Then, this 8×8 block is traversed in zig-zag like Fig. 7 :

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

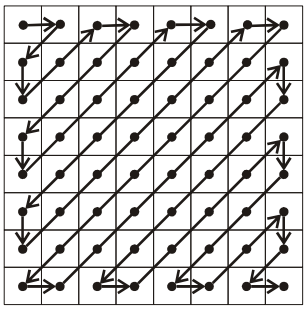


Fig. 7 Zig-Zag reordering matrix

After we are done with traversing in zig-zag the 8×8 matrix we have now a vector with 64 coefficients (0,1,...,63). The reason for this zig-zag traversing is that we traverse the 8×8 DCT coefficients in the order of increasing the spatial frequencies. So, we get a vector sorted by the criteria of the spatial frequency. In consequence in the quantized vector at high spatial frequencies, we will have a lot of consecutive zeroes.

6 Zero Run Length Coding of AC Coefficient

Now we have the quantized vector with a lot of consecutive zeroes. We can exploit this by run length coding of the consecutive zeroes. Let's consider the 63 AC coefficients in the original 64 quantized vectors first. For example, we have :

57, 45, 0, 0, 0, 0, 23, 0, -30, -16, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,..., 0

We encode for each value which is not 0, than add the number of consecutive zeroes preceding that value in front of it. The RLC (run length coding) is :

(0,57) ; (0,45) ; (4,23) ; (1,-30) ; (0,-16) ; (2,1) ; EOB

EOB (End of Block) is a special coded value. If we have reached in a position in the vector from which we have till the end of the vector only zeroes, we'll mark that position with EOB and finish the RLC of the quantized vector. Note that if the quantized vector does not finishes with zeroes (the last element is not 0), we do not add the EOB marker. Actually, EOB is equivalent to (0,0), so we have :

(0,57) ; (0,45) ; (4,23) ; (1,-30) ; (0,-16) ; (2,1) ; (0,0)

We give another example. For the quantized vector as follows :

57, eighteen zeroes, 3, 0, 0, 0, 0, 2, thirty-three zeroes, 895, EOB

The JPEG Huffman coding makes the restriction that the number of previous 0's to be coded as a 4-bit value, so it can't overpass the value 15 (0xF). So, this example would be coded as :

(0,57) ; (15,0) ; (2,3) ; (4,2) ; (15,0) ; (15,0) ; (1,895) ; (0,0)

(15,0) is a special coded value which indicates that there are 16 consecutive zeroes.

7 Difference Coding of DC Coefficients

Because the DC coefficients contains a lot of energy, it usually has much larger value than AC coefficients, and we can notice that there is a very close connection between the DC coefficients of adjacent blocks. So, the JPEG standard encode the difference between the DC coefficients of consecutive 8×8 blocks rather than its true value. The mathematical represent of the difference is :

$$Diff_i = DC_i - DC_{i-1} \tag{8}$$

and we set $DC_0 = 0$. DC of the current block DC_i will be equal to $DC_{i-1} + Diff_i$. So, in the JPEG file, the first coefficient is actually the difference of DCs as shown in Fig. 8. Then the difference is Huffman encoded together with the encoding of AC coefficients.

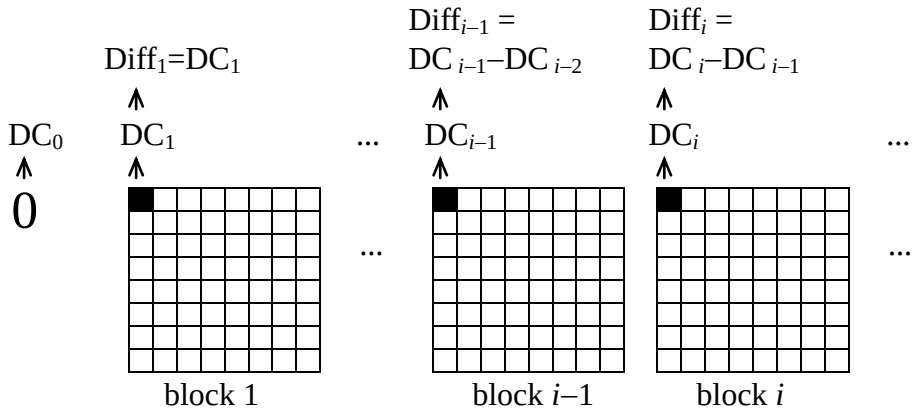


Fig. 8 Differences of 8×8 blocks

8 Huffman Coding

Instead of storing the actual value , the JPEG standard specifies that we store the minimum size in bits in which we can keep that value (it's called the category of that value) and then a bit-coded representation of that value like Figure 9 :

Category	Values	Bits for the value
1	-1,1	0,1
2	-3,-2,2,3	00,01,10,11
3	-7,-6,-5,-4,4,5,6,7	000,001,010,011,100,101,110,111
4	-15,...,-8,8,...,15	0000,...,0111,1000,...,1111
5	-31,...,-16,16,...31	00000,...,01111,10000,...,11111
6	-63,...,-32,32,...63	000000,...,011111,100000,...,111111
7	-127,...,-64,64,...,127	0000000,...,0111111,1000000,...,1111111
8	-255,..,-128,128,..,255	...
9	-511,..,-256,256,..,511	...
10	-1023,..,-512,512,..,1023	...
11	-2047,..,-1024,1024,..,2047	...

Fig. 9 Table of the category and bit-coded values

In consequence for the previous example of AC coefficients:

(0,57) ; (0,45) ; (4,23) ; (1,-30) ; (0,-8) ; (2,1) ; (0,0)

We encode only the right value of these pairs as category and bits for the value, except the pairs that are special markers like (0,0) or (15,0). For example, 57 is in the category 6 and it is bit-coded 111001, so we will encode it as 6,111001. And we write again the string of pairs:

(0,6,111001) ; (0,6,101101) ; (4,5,10111); (1,5,00001) ; (0,4,0111) ; (2,1,1) ; (0,0)

The first 2 values in bracket paranthesis can be represented on a byte because of the fact that each of the 2 values is 0,1,2,...,15. In this byte, the higher 4-bit represents the number of previous 0s, and the lower 4-bit is the category of the value which is not 0.

run/category	code length	code word
0/0	4	1010
...		
0/6	7	1111000
...		
0/10	16	1111111110000011
1/1	4	1100
...		
4/5	16	1111111110011000
...		
15/10	16	1111111111111110

Fig. 10 Huffman table of luminance AC coefficients

The final step is encoding this byte using Huffman coding. For example, if the Huffman code of byte (0,6) is 111000, and the Huffman code of byte (4,5) is 1111111110011001, and so on. The final stream of bits written in the JPEG file on disk for the previous example of 63 coefficients is :

1111000 1111001 , 111000 101101 , 1111111110011000 10111 ,
11111110110 00001 , 1011 0111 , 11100 1 , 1010

category	code length	code word
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110

Fig. 11 Huffman table of luminance DC coefficients

Now we cosider the encoding of the difference of DC coefficients. The difference will be represented by category and its bits for the value, and it will be Huffman encoded only the category value. For example, if difference is equal to -511, then it will be represented as (9,000000000). If the Huffman code of 9 is 1111110, the stream of bits written in the JPEG file on disk for the difference is :

1111110 000000000

Finally, we combine this example of DC and to the previous example of ACs, for this vector with 64 coefficients, the final stream of bits written in the JPEG file will be :

1111110 000000000 , 1111000 1111001 , 111000 101101 ,
1111111110011000 10111 , 11111110110 00001 , 1011 0111 , 11100 1 , 1010

9 Conclusions

We have introduced the basic compression methods of JPEG standard. Although this standard has become the most popular image format, it still has some properties to improvement. For example, the new JPEG 2000 standard use wavelet-based compression method, and it can operate at higher compression ratio without generating the characteristic 'blocky and blurry' artifacts of the original DCT-based JPEG standard.

References

[1] 酒井善則、吉田俊之 共著，白執善 編譯，“影像壓縮技術”，全華，2004

[2] T. Acharya, A. K. Ray, "Image Processing: Principles and Applications", John Wiley & Sons, 2005, pp.351-368.

[3] R. C. Gonzolez, R. E. Woods, S. L. Eddins, "Digital Image Processing Using Matlab", Prentice Hall, 2004.

[4] G. K. Wallace, 'The JPEG Still Picture Compression Standard', Communications of the ACM, Vol. 34, Issue 4, pp.30-44.

[5] C. Cuturicu, 'A note about the JPEG decoding algorithm', available in http://www.opennet.ru/docs/formats/jpeg.txt, 1999.

[6] ITU-T Recommendation T.81, 'Digital compression and coding of continuous-tone still images - Requirements and guidelines', available in http://www.itu.int/rec/T-REC-T/e

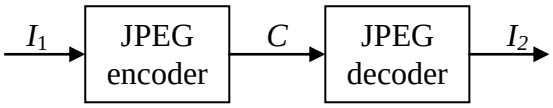
[7] The Independent JPEG Group, C source code of JPEG Encoder research 6b, 1998.

Simulation Results

I_1 : Original image with width W and height H

C : Encoded jpeg stream from I_1

I_2 : Decoded image from C



Compression ratio = sizeof(C) / sizeof(I_1)

Root mean square error = $\sqrt{\sum_{y=1}^{H}\sum_{x=1}^{W}\left[I_1(x,y)-I_2(x,y)\right]^2/(H\times W)}$

Quantization : 1 , Downsampling : 444	Quantization : 1 , Downsampling : 420
Compression ratio : 0.084524 Root mean square error : 6.83	Compression ratio : 0.059011 Root mean square error : 8.08
Quantization : 4 , Downsampling : 444	Quantization : 4 , Downsampling : 420
Compression ratio : 0.036143 Root mean square error : 11.56	Compression ratio : 0.027873 Root mean square error : 12.46

From the table above and the figure next page, we can see that the type 420 of downsampling can decrease the compression ratio with little vision error, and the quantization level 4 can also decrease the compression ratio but generating the clear characteristic 'blocky and blurry' artifacts.

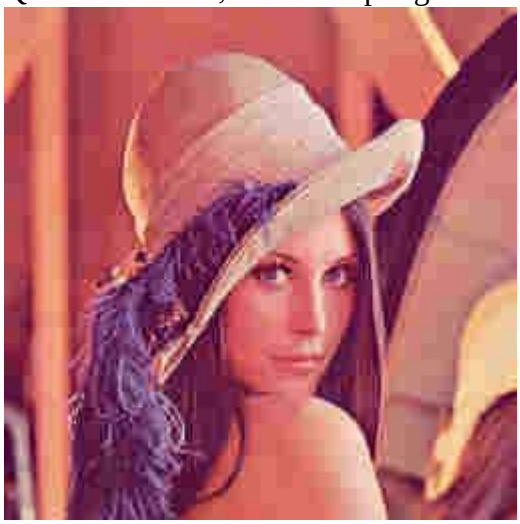
Original image : lena.bmp

Compression image :

Quantization : 1 , Downsampling : 444

Quantization : 1 , Downsampling : 420

Quantization : 4 , Downsampling : 444



Quantization : 4 , Downsampling : 420