Basic Image Compression Algorithm and Introduction to JPEG Standard

Pao-Yen Lin

E-mail: r97942117@ntu.edu.tw

Graduate Institute of Communication Engineering

National Taiwan University, Taipei, Taiwan, ROC

Abstract

Because of the explosively increasing information of image and video in various storage devices and Internet, the image and video compression technique becomes more and more important. This paper introduces the basic concept of data compression which is applied to modern image and video compression techniques such as JPEG, MPEG, MPEG-4 and so on.

The basic idea of data compression is to reduce the data correlation. By applying Discrete Cosine Transform (DCT), the data in time (spatial) domain can be transformed into frequency domain. Because of the less sensitivity of human vision in higher frequency, we can compress the image or video data by suppressing its high frequency components but do no change to our eye.

Moving pictures such as video are data in three-dimensional space consists of spatial plane and time axis. Therefore, in addition to reducing spatial correlation, we need to reduce the time correlation. We introduce a method called Motion Estimation (ME). In this method, we find similar part of image in previous or future frames. Then replace the image by a Motion Vector (MV) in order to reduce time correlation.

In this paper, we also introduce JPEG standard and MPEG standard which are the well-known image and video compression standard, respectively.

1 Introduction

Nowadays, the size of storage media increases day by day. Although the largest capacity of hard disk is about two Terabytes, it is not enough large if we storage a video file without compressing it. For example, if we have a color video file stream, that is, with three 720x480 sized layer, 30 frames per second and 8 bits for each pixel. Then we need $720 \times 480 \times 3 \times 8 \times 30 \cong 249\text{Mbit/s}$! This equals to about 31.1MB per second. For a 650MB CD-ROM, we can only storage a video about 20 seconds long. That is why we want to do image and video compression though the capacity of storage media is quite large now.

In chapter 2, we will introduce the basic concept of data compression. The main idea of data compressing is reducing the data correlation and replacing them with simpler data form. Then we will discuss the method that is common used in image/video compression in chapter 3.

In chapter 4 and chapter 5, we will introduce quantization and entropy coding. After reducing data correlation, the amounts of data are not really reduced. We use quantization and entropy coding to compress the data.

In chapter 6, we give an example of image compression − JPEG standard. The JPEG standard has been widely used in image and photo compression recently.

In chapter 7, we discuss how to reduce time correlation with a method called Motion Estimation (ME). And then we give an example of video compression − MPEG standard in chapter 8.

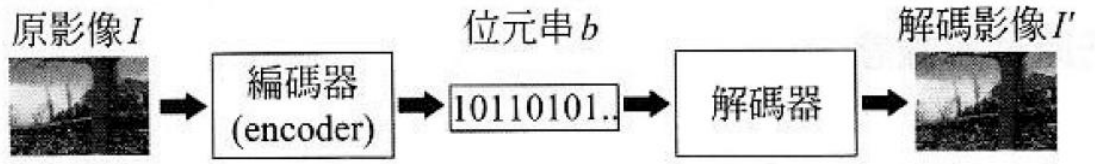

Fig. 1 Encoder and decoder of images from Ref. [3]

2 Basic Concept of Data Compression

The motivation of data compression is using less quantity of data to represent the original data without distortion of them. Consider the system in **Fig. 1**, when the

(a) Original image 83261bytes (b) Decoded image 15138bytes

Fig. 2 Example of image compression using JPEG standard

encoder receives the target image, it converts the image into bit stream b. On the other hand, the decoder receives the bit stream and then converts it back to the image I'. If the quantity of bit stream b less than the original image then we call this process *Image Compression Coding*.

There is an example in **Fig. 2** using JPEG image compression standard. The compression ratio is $15138/83261$, about 0.1818, around one fifth of the original size. Besides, we can see that the decoded image and the original image are only slightly different. In fact, the two images are not completely same, that is, parts of information are lost during the image compression process. For this reason, the decoder cannot rebuild the image perfectly. This kind of image compression is called *non-reversible coding* or *lossy coding*. On the contrary, there is another form called *reversible coding* that can perfectly rebuild the original image without any distortion. But the compression ratio of reversible coding is much lower.

For lossy coding, there is a distortion between the original image and the decoded image. In order to evaluate the coding efficiency, we need a method to evaluate the degree of distortion. There are two common evaluation tools, which are *Mean Square Error* (MSE) and *Peak Signal to Noise Ratio* (PSNR). They are defined as following:

$$\mathrm{MSE} = \sqrt{\frac{\sum_{x=0}^{W-1}\sum_{y=0}^{H-1}\left(f(x,y)-f'(x,y)\right)^2}{WH}} \tag{1}$$

$$\mathrm{PSNR} = 20\log_{10}\frac{255}{\mathrm{MSE}} \tag{2}$$

See **Eq. (1)**, $f(x,y)$ and $f'(x,y)$ denote the original image and decoded image, respectively. The image size is $W \times H$. In **Eq. (2)**, the PSNR formula is common used for 8-bits image. Note that the larger the PSNR, that smaller the degree of distortion.

Now, we want to know how and why we can make an image compressed. Generally speaking, the neighboring pixels in an image have highly correlation to each other. That is why images can be compressed in a high compression ratio.

The image coding algorithm today consists of reducing correlation between pixels, quantization and entropy coding. We will discuss these parts one by one in the following chapters. The coding algorithm system model is shown in **Fig. 3**.

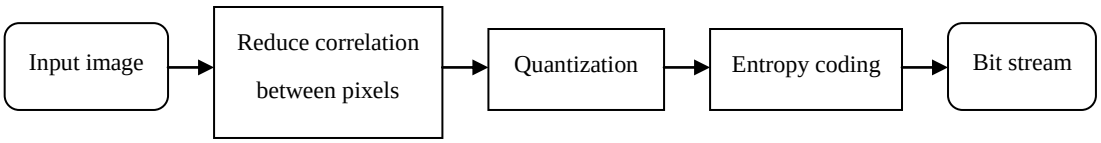

Fig. 3 general constitution of image coding algorithm

3 Orthogonal Transform and Discrete Cosine Transform

3.1 Linear transformation

We have studied linear transformation in Linear Algebra. It is very useful to represent signals in basis form. For simpleness, we discuss the case in three dimensional space whereas the case in N dimensional space can be derive easily in the same concept. We can express any three dimensional vector $\mathbf{x}$ in a column vector $[x_1, x_2, x_3]^t$, where x_1, x_2 and x_3 are values of the three corresponding axes. For a proper transformation matrix $\mathbf{A}$, we can transform vector $\mathbf{x}$ into another vector $\mathbf{y}$, we call this a *linear transformation* process. It can be written as:

$$\mathbf{y} = \mathbf{A}\mathbf{x} \tag{3}$$

where $\mathbf{x}$ and $\mathbf{y}$ are vectors in $\Re^3$ space and $\mathbf{A}$ is called a transformation matrix. Moreover, consider three linear independent vectors with different direction:

$$v_1 = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^t, v_2 = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}^t, v_3 = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}^t \tag{4}$$

Then, any vector in the $\Re^3$ space can be expressed as the combination of these three independent vectors, that is,

$$\mathbf{x} = a_1\mathbf{v_1} + a_2\mathbf{v}_2 + a_3\mathbf{v}_3 \tag{5}$$

where a_1, a_2 and a_3 are constants.

3.2 Orthogonal Transformation

According to **3.3**, any vector in the $\Re^3$ space can be expressed as the combination of three independent vectors. If we choose these three independent vectors such that they are mutually independent, we will have many useful properties and the numerical computation will become easier. As the same in **Eq. (5)**, moreover, we will have $\mathbf{v_1}, \mathbf{v_2}$ and $\mathbf{v_3}$ that satisfy

$$\begin{cases} \mathbf{v_1} \cdot \mathbf{v}_2 = \mathbf{v}_2 \cdot \mathbf{v}_3 = \mathbf{v_1} \cdot \mathbf{v}_3 = 0 \\ \mathbf{v_1} \cdot \mathbf{v_1} = \left|\mathbf{v_1}\right|^2 = 1 \\ \mathbf{v}_2 \cdot \mathbf{v}_2 = \left|\mathbf{v}_2\right|^2 = 1 \\ \mathbf{v}_3 \cdot \mathbf{v}_3 = \left|\mathbf{v}_3\right|^2 = 1 \end{cases} \tag{6}$$

From **Eq. (5)** and **Eq. (6)**, a_1, a_2 and a_3 can be found by

$$a_1 = \mathbf{x} \cdot \mathbf{v_1},\ a_2 = \mathbf{x} \cdot \mathbf{v}_2,\ a_3 = \mathbf{x} \cdot \mathbf{v}_3 \tag{7}$$

We find that it is easy to obtain a_1, a_2 and a_3 just by taking inner product of the vector $\mathbf{x}$ and corresponding vectors.

3.2 Karhunen-Loeve Transformation

Because images have high correlation in a small area, for an image with size $K_1 \times K_2$, we usually divide it into several small blocks with size $N_1 \times N_2$ and we deal with each block with a transformation that can reduce its pixel correlation separately. This can be seen in **Fig. 4**. Moreover, if we choose bigger block size we may obtain higher compression ratio. However, an oversized block size may have lower pixel correlation. There will be a tradeoff.

In order to do linear transformation to each block in the image, we may scan the pixel in the transformation blocks and transform it into an N dimensional vector. This can be seen in **Fig. 5**. The number of total transformation blocks equals to $M \equiv K_1 K_2 / N_1 N_2$ and the number of pixels in a transformation block is $N \equiv N_1 N_2$. After horizontal scanning, we have M vectors:

$$\begin{cases} \mathbf{x}^{(1)} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_N^{(1)} \end{bmatrix}^t \\ \vdots \\ \mathbf{x}^{(m)} = \begin{bmatrix} x_1^{(m)} & x_2^{(m)} & \cdots & x_N^{(m)} \end{bmatrix}^t \\ \vdots \\ \mathbf{x}^{(M)} = \begin{bmatrix} x_1^{(M)} & x_2^{(M)} & \cdots & x_N^{(M)} \end{bmatrix}^t \end{cases} \tag{8}$$

What we want the do is to achieve the optimal orthogonal transform for these vectors in order to reduce the pixel correlation in each transformation blocks. That is, find a transformation matrix $\mathbf{V}$ such that

$$\mathbf{y}^{(m)} = \mathbf{V}^t \mathbf{x}^{(m)} \tag{9}$$

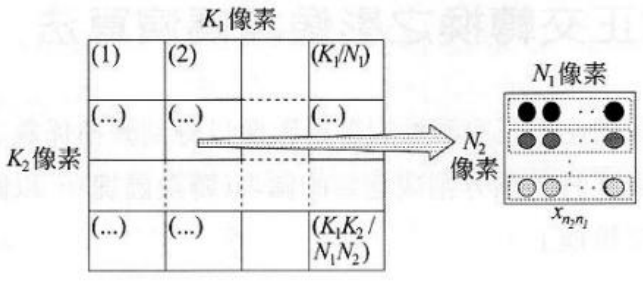

Fig. 4 Image partition and transformation block Ref. [3]

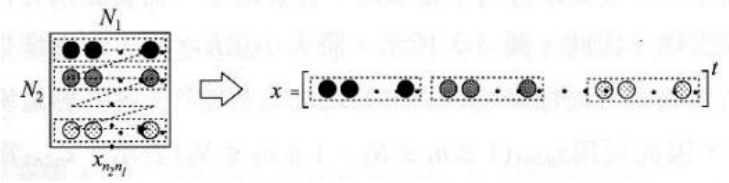

Fig. 5 Transform a transformation block into an N dimensional vector Ref. [3]

Now, consider the covariance of these vectors

$$C_{x_i x_j} = E\left[\left(x_i^{(m)} - \overline{x}_i^{(m)}\right)\left(x_j^{(m)} - \overline{x}_j^{(m)}\right)\right] \tag{10}$$

$$C_{y_i y_j} = E\left[\left(y_i^{(m)} - \overline{y}_i^{(m)}\right)\left(y_j^{(m)} - \overline{y}_j^{(m)}\right)\right] \tag{11}$$

The *i*-th element of $\mathbf{y}^{(m)}$ can be written as

$$y_i^{(m)} = \sum_{n=1}^{N} v_{ni} x_n^{(m)} \tag{12}$$

Then the mean of $\mathbf{y}^{(m)}$ is

$$\overline{y}_i^{(m)} = E\left[\sum_{n=1}^{N} v_{ni} x_n^{(m)}\right] = \sum_{n=1}^{N} v_{ni} E\left[x_n^{(m)}\right] = \sum_{n=1}^{N} v_{ni} \overline{x}_n^{(m)} \tag{13}$$

For simpleness, we assume each pixel value $x_i^{(m)}$ is subtracted with its mean value, that is, we substitute $x_i^{(m)}$ with $x_i^{(m)} - \overline{x}_i^{(m)}$. Then the means of latest pixel value $x_i^{(m)}$ and $y_i^{(m)}$ change to zero. Thus we can rewrite **Eq. (10)** and **Eq. (11)**:

$$C_{x_i x_j} = E\left[x_i^{(m)} x_j^{(m)}\right] \tag{14}$$

$$C_{y_i y_j} = E\left[y_i^{(m)} y_j^{(m)}\right] \tag{15}$$

Eq. (14) and Eq.(15) can be written in a matrix form:

$$C_{xx} = \begin{bmatrix} E\left[x_1^{(m)} x_1^{(m)}\right] & \cdots & E\left[x_1^{(m)} x_N^{(m)}\right] \\ \vdots & \ddots & \vdots \\ E\left[x_N^{(m)} x_1^{(m)}\right] & \cdots & E\left[x_N^{(m)} x_N^{(m)}\right] \end{bmatrix} \tag{16}$$

$$C_{yy} = \begin{bmatrix} E\left[y_1^{(m)} y_1^{(m)} \right] & \cdots & E\left[y_1^{(m)} y_N^{(m)} \right] \\ \vdots & \ddots & \vdots \\ E\left[y_N^{(m)} y_1^{(m)} \right] & \cdots & E\left[y_N^{(m)} y_N^{(m)} \right] \end{bmatrix} \tag{17}$$

These are called a covariance matrix. We can easily find that it must be a symmetric matrix. They can be rewritten in a vector form:

$$C_{xx} = E\left[\mathbf{x}^{(m)} \left(\mathbf{x}^{(m)} \right)^t \right] \tag{18}$$

$$C_{yy} = E\left[\mathbf{y}^{(m)} \left(\mathbf{y}^{(m)} \right)^t \right] \tag{19}$$

Moreover, $\mathbf{y}^{(m)}$ is obtained by applying linear transformation matrix $\mathbf{V}$ on $\mathbf{x}^{(m)}$:

$$\mathbf{y}^{(m)} = \mathbf{V}^t \mathbf{x}^{(m)} \tag{20}$$

By **Eq. (19)** and **Eq. (20)**, we find that

$$C_{yy} = E\left[\mathbf{V}^t \mathbf{x}^{(m)} \left(\mathbf{V}^t \mathbf{x}^{(m)} \right)^t \right] = \mathbf{V}^t E\left[\mathbf{x}^{(m)} \left(\mathbf{x}^{(m)} \right)^t \right] \mathbf{V} = \mathbf{V}^t C_{xx} \mathbf{V} \tag{21}$$

The purpose is to obtain uncorrelated $\mathbf{y}^{(m)}$. Note that for an uncorrelated $\mathbf{y}^{(m)}$, it has a covariance matrix C_{yy} which is a diagonal matrix. From **Eq. (21)**, we find that if C_{yy} is a diagonal matrix then we can regard the right-hand part of this equation is the eigenvalue-eigenvector decomposition of C_{xx}. The matrix $\mathbf{V}$ is composed of the eigenvectors of C_{xx}. Usually, we have it ordered with eigenvectors that have bigger corresponding eigenvalues to smaller ones. This is called the Karhunen-Loeve Transform (KLT).

3.3 Discrete Cosine Transform

For common used such as JPEG standard or MPEG standard, we do not use KLT.

Although we can have the optimal orthogonal transformation by applying KLT, it still has the following drawbacks:

1 Each image has to do the KLT respectively. This makes the computation complexity large.

2 In order to decode the encoded image we have to transmit the KLT transformation matrix to the decoder. It costs another process time and memory spaces.

Therefore, if we can derive an orthogonal transform that can preserve the optimal property of KLT for all image then we can deal with the problems we mentioned. Then we have the Discrete Cosine Transform (DCT). The forward DCT is defined as

$$F(u,v)=\frac{1}{4}C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7}f(x,y)\cos\left[\frac{\pi(2x+1)u}{16}\right]\cos\left[\frac{\pi(2y+1)v}{16}\right]$$
$$\text{for } u=0,...,7 \text{ and } v=0,...,7 \qquad (22)$$
$$\text{where } C(k)=\begin{cases}1/\sqrt{2} & \text{for } k=0\\ 1 & \text{otherwise}\end{cases}$$

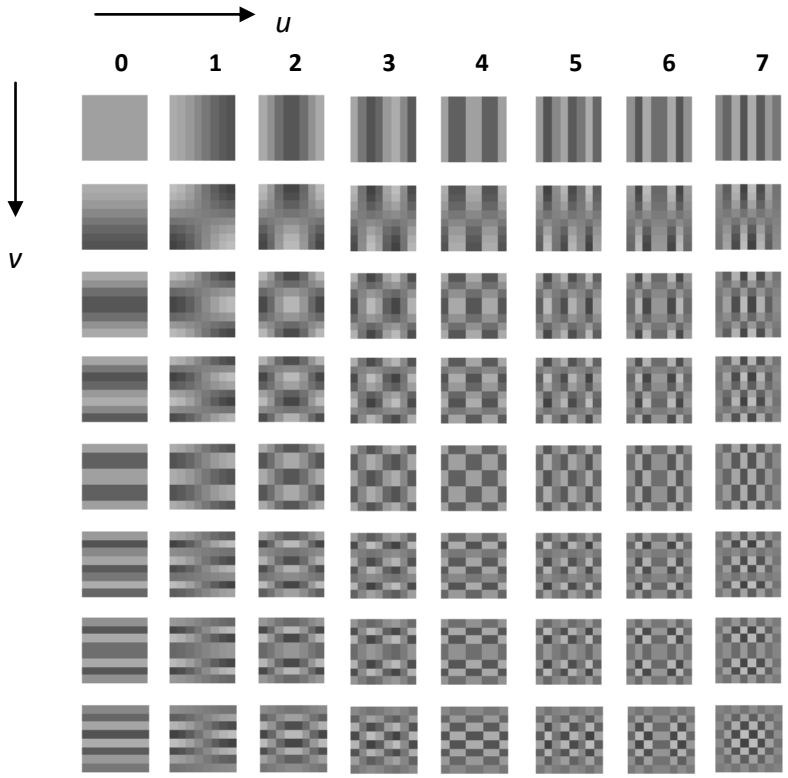

Fig. 6 The 8×8 DCT basis $\omega_{x,y}(u,v)$

And the inverse DCT is defined as the following equation:

$$f(x,y)=\frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7}C(u)C(v)F(u,v)\cos\left[\frac{\pi(2x+1)u}{16}\right]\cos\left[\frac{\pi(2y+1)v}{16}\right] \quad (23)$$

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

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

$$\omega_{x,y}(u,v)=\frac{C(u)C(v)}{4}\cos\left[\frac{\pi(2x+1)u}{16}\right]\cos\left[\frac{\pi(2y+1)v}{16}\right] \quad (24)$$

Then we can rewrite the IDCT by **Eq. (24)**:

$$f(x,y)=\sum_{u=0}^{7}\sum_{v=0}^{7}F(u,v)\omega_{x,y}(u,v) \quad \text{for } x=0,...,7 \text{ and } y=0,...,7 \quad (25)$$

The 8×8 two dimensional DCT basis is depicted in **Fig. 6**.

4 Quantization

The transformed 8×8 block in **Fig. 6** 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 **Eq. (22)** for $F(u,v)$.

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

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

Fig. 7 Quantization matrix

The next step in the compression process is to quantize the transformed coefficients. Each of the 64 DCT coefficients is 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 quantize step-size parameter $Q(u,v)$ in the quantization matrix and rounded to the nearest integer as :

$$F_q(u,v) = Round\left(\frac{F(u,v)}{Q(u,v)}\right) \tag{26}$$

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 in **Fig. 7**.

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 An Example of Image Compression−JPEG Standard

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 has two basic compression methods. The DCT-based method is specified for lossy compression, and the predictive method is specified for lossless compression. In this article, we will introduce the lossy compression of JPEG standard. **Fig. 8** shows the block diagram of Baseline JPEG encoder.

5.1 Zig-zag Reordering

After doing 8×8 DCT and quantization over a block we have new 8×8 blocks which denotes the value in frequency domain of the original blocks. Then we have to reorder the values into one dimensional form in order to encode them. The DC coefficient is encoded by difference coding. It will be discussed later. However, the AC terms are scanned in a Zig-zag manner. 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. The Zig-zag reordering process is shown in **Fig. 9**.

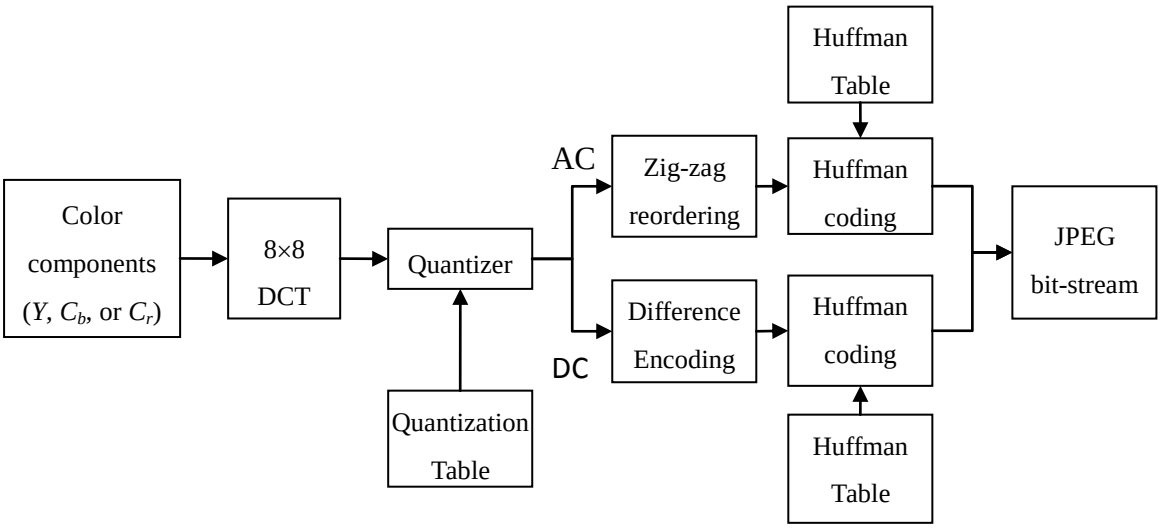

Fig. 8 Baseline JPEG encoder

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

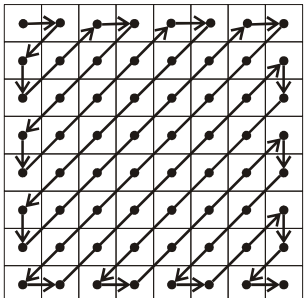

Fig. 9 Zig-Zag reordering matrix

5.2 Zero Run Length Coding of AC Coefficient

Now we have the one dimensional quantized vector with a lot of consecutive zeroes. We can process 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

The 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)

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.

5.3 Difference Coding of DC Coefficient

Because the DC coefficients in the blocks are highly correlated with each other. Moreover, DC coefficients contain a lot of energy so they usually have much larger value than AC coefficients. That is why we have to reduce the correlation before doing encoding. The JPEG standard encodes the difference between the DC coefficients. We compute the difference value between adjacent DC values by the following equation:

$$Diff_i = DC_i - DC_{i-1} \tag{27}$$

Note that the initial DC value is set to zero. Then the difference is Huffman encoded together with the encoding of AC coefficients. The difference coding process is shown in **Fig. 10**.

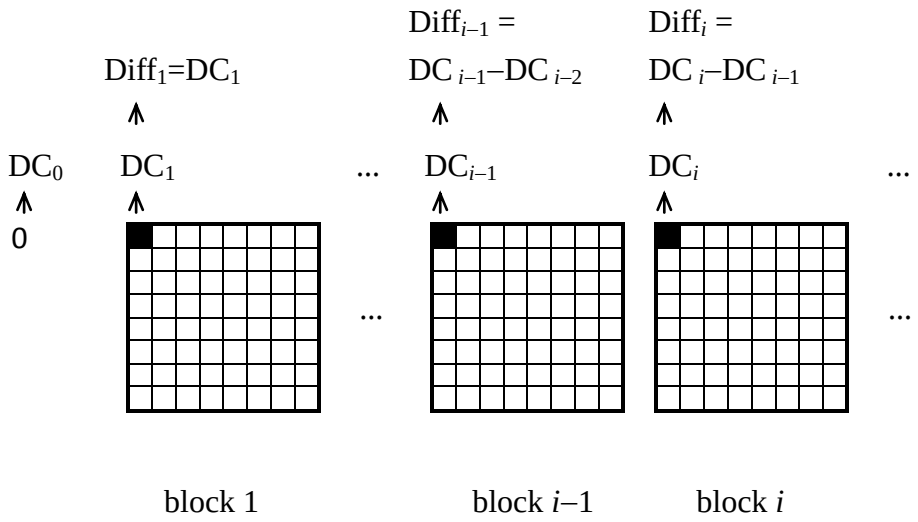

Fig. 10 Difference coding of DC coefficients

Because the Huffman coding is not in the scope of this research, the Huffman coding is not discussed in this paper.

6 Conclusions

We have introduced the basic concepts of image compression and the overview of JPEG standard. Although there is much more details we did not mentioned, the important parts are discussed in this paper. The JPEG standard has become the most popular image format; it still has some properties to improvement. The compression

ratio can be higher without block effect by using wavelet-based JPEG 2000 standard.

7 References

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing 2/E*. Upper Saddle River, NJ: Prentice-Hall, 2002.

[2] J. J. Ding and J. D. Huang, “Image Compression by Segmentation and Boundary Description,” June, 2008.

[3] 酒井善則・吉田俊之 共著，白執善 編譯，影像壓縮技術. 全華科技圖書，2004年10月.

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