JPEG Compression:

- JPEG : Joint Photographic Experts Group
- The first international static image compression standard Published in 1992.
- The main reason for JPEG success is the quality of its output for relatively good compression ratio.
- JPEG is a lossy image compression method. It employs a transform coding method using the DCT (Discrete Cosine Transform).
- An image is a function of i and j (or conventionally x and y) in the spatial domain. The 2D DCT is used as one step in JPEG **in order to yield a frequency response** which is a **function F(u, v)** in the spatial frequency domain, indexed by two integers **u and v**.

Observations for JPEG Image Compression:

The effectiveness of the DCT transform coding method in JPEG relies on 3 major observations:

- **Observation 1:** Useful image contents change relatively slowly across the image, i.e., it is unusual for intensity values to vary widely several times in a small area, for example, within an 8×8 image block. much of the information in an image is repeated, hence "spatial redundancy".
- **Observation 2:** Psychophysical experiments suggest that humans are much less likely to notice the loss of very high spatial frequency components than the loss of lower frequency components. The spatial redundancy can be reduced by largely reducing the high spatial frequency contents.

- **Observation 3:** Visual acuity (accuracy in distinguishing closely spaced lines) is much greater for gray ("black and white") than for color.

Main Steps in JPEG Image Compression:

1. Transform RGB to YIQ or YUV and subsample color
2. Perform DCT on image blocks
3. Apply Quantization
4. Zigzag Ordering
5. DPCM on DC coefficients
6. RLE on AC coefficients
7. Perform entropy coding

The figure below shows the block diagram for JPEG encoder. If we reverse the arrows in the figure, we basically obtain a JPEG decoder.

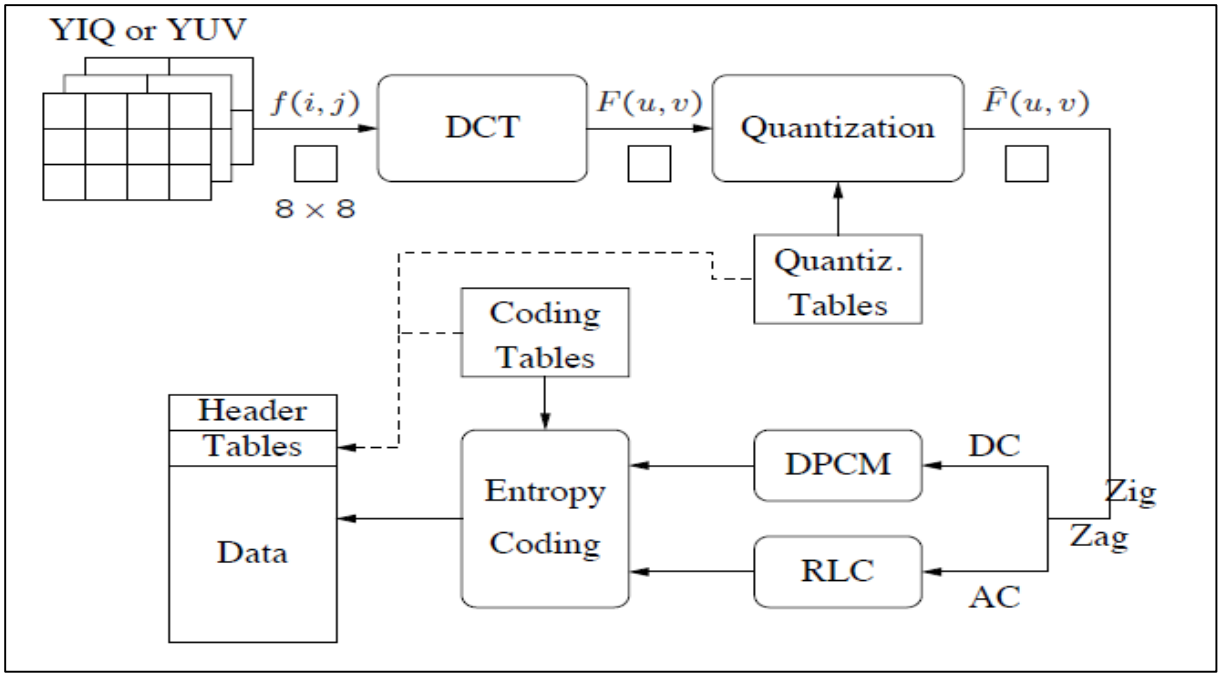


Figure (2) Block Diagram of JPEG Encoder

1. Optional Converting RGB to YUV

• YUV color mode stores color in terms of its luminance (brightness) and chrominance (hue).

• The human eye is less sensitive to chrominance than luminance.

• YUV is not required for JPEG compression, but it gives a better compression rate.

RGB vs. YUV

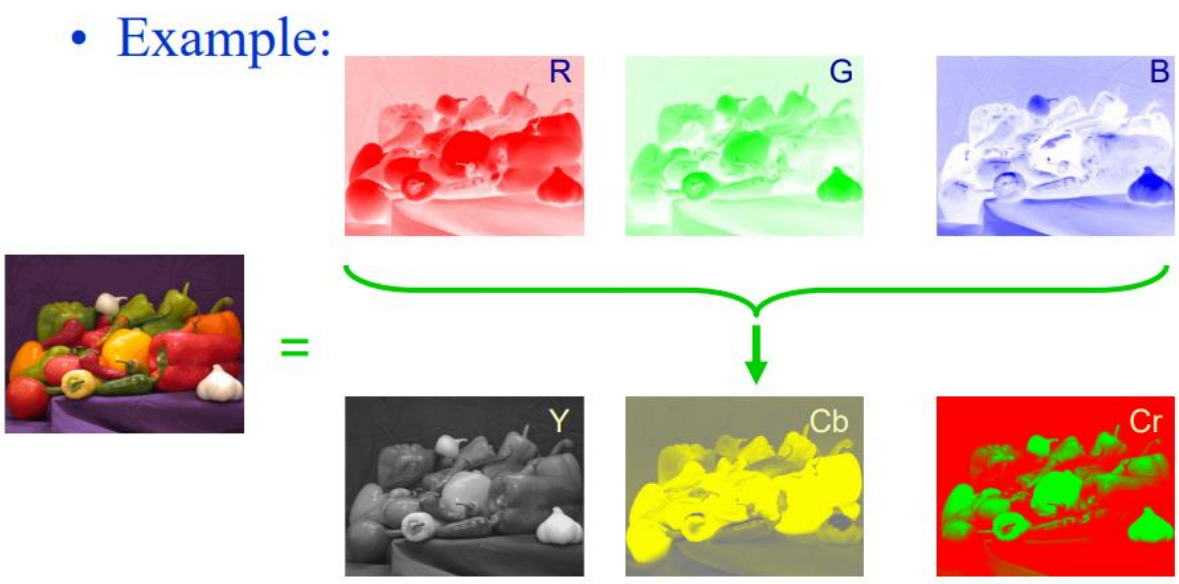
• It's simple arithmetic to convert RGB to YUV. The formula is based on the relative contributions that red, green, and blue make to the luminance and chrominance factors.

• There are several different formulas in use depending on the target monitor. For example:

$$Y = 0.299 * R + 0.587 * G + 0.114 * B$$

$$U = -0.1687 * R – 0.3313* G + 0.5 * B + 128$$

$$V = 0.5 * R – 0.4187 * G – 0.813 * B + 128$$



Remember: all JPEG process is operating on YCbCr color space !

2. DCT on image blocks

Each image is divided into 8 × 8 blocks each is called ***data unit***. The 2D DCT is applied to each block image $f(i, j)$, with output being the DCT coefficients $F(u, v)$ for each block. They represent the average pixel value and successive higher-frequency changes within the group. This prepares the image data for the crucial step of losing information. ***Since DCT involves the transcendental function cosine, it must involve some loss of information due to the limited precision of computer arithmetic.*** This means that even without the main lossy step (the quantization step), there will be some loss of image quality, but it is normally small. As an example, one such 8×8 8-bit subimage might be:

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

Before computing the DCT of the 8×8 block, its values are shifted from a positive range to one centered on zero. For an 8-bit image, each entry in the original block falls in the range {\displaystyle [0,255]}. The midpoint of the range (in this case, the value 128) is subtracted from each entry to produce a data range that is centered on zero, so that the modified range is {\displaystyle [-128,127]}. This step reduces the dynamic range requirements in the DCT processing stage that follows. This step results in the following values:

$$g = \begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

($x \longrightarrow$, $\downarrow y$.)

The next step is to take the two-dimensional DCT, which is given by:

The next step is to take the two-dimensional DCT, which is given by:

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

where

- u is the horizontal spatial frequency, for the integers $0 \leq u < 8$.
- v is the vertical spatial frequency, for the integers $0 \leq v < 8$.
- $\alpha(u) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } u = 0 \\ 1, & \text{otherwise} \end{cases}$ is a normalizing scale factor to make the transformation orthonormal
- $g_{x,y}$ is the pixel value at coordinates (x, y)
- $G_{u,v}$ is the DCT coefficient at coordinates (u, v).

If we perform this transformation on our matrix above, we get the following (rounded to the nearest two digits beyond the decimal point):

$$G = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \downarrow v. \end{matrix}$$

$u \longrightarrow$

3. Quantization

Each of the 64 frequency components in a data unit is divided by a separate number called its ***quantization coefficient*** (QC), and then rounded to an integer. This is where information is irretrievably lost. Large QCs cause more loss, so the high frequency components typically have larger QCs. Each of the 64 QCs is a JPEG parameter and can, in principle, be specified by the user. In practice, most JPEG implementations use the QC tables recommended by the JPEG standard for the ***luminance*** and ***chrominance*** image components.

$$\hat{F}(u,v) = round\left(\frac{F(u,v)}{Q(u,v)}\right)$$

Below is an example of the jpge compression for a smooth image block

Luminance Quantization Table

```
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
```

Chrominance Quantization Table

```
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
```

The human eye is good at seeing small differences in brightness over a relatively large area, but not so good at distinguishing the exact strength of a high frequency brightness variation. This allows one to greatly reduce the amount of information in the high frequency components. This is done by simply dividing each component in the frequency domain by a constant for that component, and then rounding to the nearest integer. This rounding operation is the only lossy operation in the whole process (other than chroma subsampling) if the DCT computation is performed with sufficiently high precision. As a result of this, it is typically the case that many of the higher frequency components are rounded to zero, and many of the rest become small positive or negative numbers, which take many fewer bits to represent.

The elements in the quantization matrix control the compression ratio, with larger values producing greater compression. A typical quantization matrix (for a quality of 50% as specified in the original JPEG Standard), is as follows:

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

The quantized DCT coefficients are computed with

$$B_{j,k} = \text{round}\left(\frac{G_{j,k}}{Q_{j,k}}\right) \text{ for } j = 0, 1, 2, \ldots, 7; k = 0, 1, 2, \ldots, 7$$

$$B_{j,k} = \text{round}\left(\frac{G_{j,k}}{Q_{j,k}}\right) \text{ for } j = 0, 1, 2, \ldots, 7; k = 0, 1, 2, \ldots, 7$$

where G is the unquantized DCT coefficients; Q is the quantization matrix above; and B is the quantized DCT coefficients.

Using this quantization matrix with the DCT coefficient matrix from above results in:

$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

For example, using −415 (the DC coefficient) and rounding to the nearest integer

$$\text{round}\left(\frac{-415.37}{16}\right) = \text{round}\left(-25.96\right) = -26.$$

Notice that most of the higher-frequency elements of the sub-block (i.e., those with an *x* or *y* spatial frequency greater than 4) are compressed into zero values.

An 8×8 block from the Y image of 'Lena'

200	202	189	188	189	175	175	175
200	203	198	188	189	182	178	175
203	200	200	195	200	187	185	175
200	200	200	200	197	187	187	187
200	205	200	200	195	188	187	175
200	200	200	200	200	190	187	175
205	200	199	200	191	187	187	175
210	200	200	200	188	185	187	186

$f(i,j)$

515	65	-12	4	1	2	-8	5
-16	3	2	0	0	-11	-2	3
-12	6	11	-1	3	0	1	-2
-8	3	-4	2	-2	-3	-5	-2
0	-2	7	-5	4	0	-1	-4
0	-3	-1	0	4	1	-1	0
3	-2	-3	3	3	-1	-1	3
-2	5	-2	4	-2	2	-3	0

$F(u,v)$

32	6	-1	0	0	0	0	0
-1	0	0	0	0	0	0	0
-1	0	1	0	0	0	0	0
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$\hat{F}(u,v)$

512	66	-10	0	0	0	0	0
-12	0	0	0	0	0	0	0
-14	0	16	0	0	0	0	0
-14	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$\tilde{F}(u,v)$

199	196	191	186	182	178	177	176
201	199	196	192	188	183	180	178
203	203	202	200	195	189	183	180
202	203	204	203	198	191	183	179
200	201	202	201	196	189	182	177
200	200	199	197	192	186	181	177
204	202	199	195	190	186	183	181
207	204	200	194	190	187	185	184

$\tilde{f}(i,j)$

1	6	-2	2	7	-3	-2	-1
-1	4	2	-4	1	-1	-2	-3
0	-3	-2	-5	5	-2	2	-5
-2	-3	-4	-3	-1	-4	4	8
0	4	-2	-1	-1	-1	5	-2
0	0	1	3	8	4	6	-2
1	-2	0	5	1	1	4	-6
3	-4	0	6	-2	-2	2	2

$\epsilon(i,j) = f(i,j) - \tilde{f}(i,j)$

Notes that changing the compression ratio simply by multiplicatively scaling the numbers in the *Q(u,v)* matrix. In fact, the *quality factor*, a user choice offered in every JPEG implementation, is essentially linearly tied to the scaling factor.

Now note how the $\boldsymbol{\epsilon(i,j) = f(i,j) - \tilde{f}(i,j)}$ is differ in the examples above, why? In the first example the pixel values in the block contain few high –spatial frequency changes. i.e. JPEG dose introduce more loss if the image has quickly changing details.