JPEG Compression

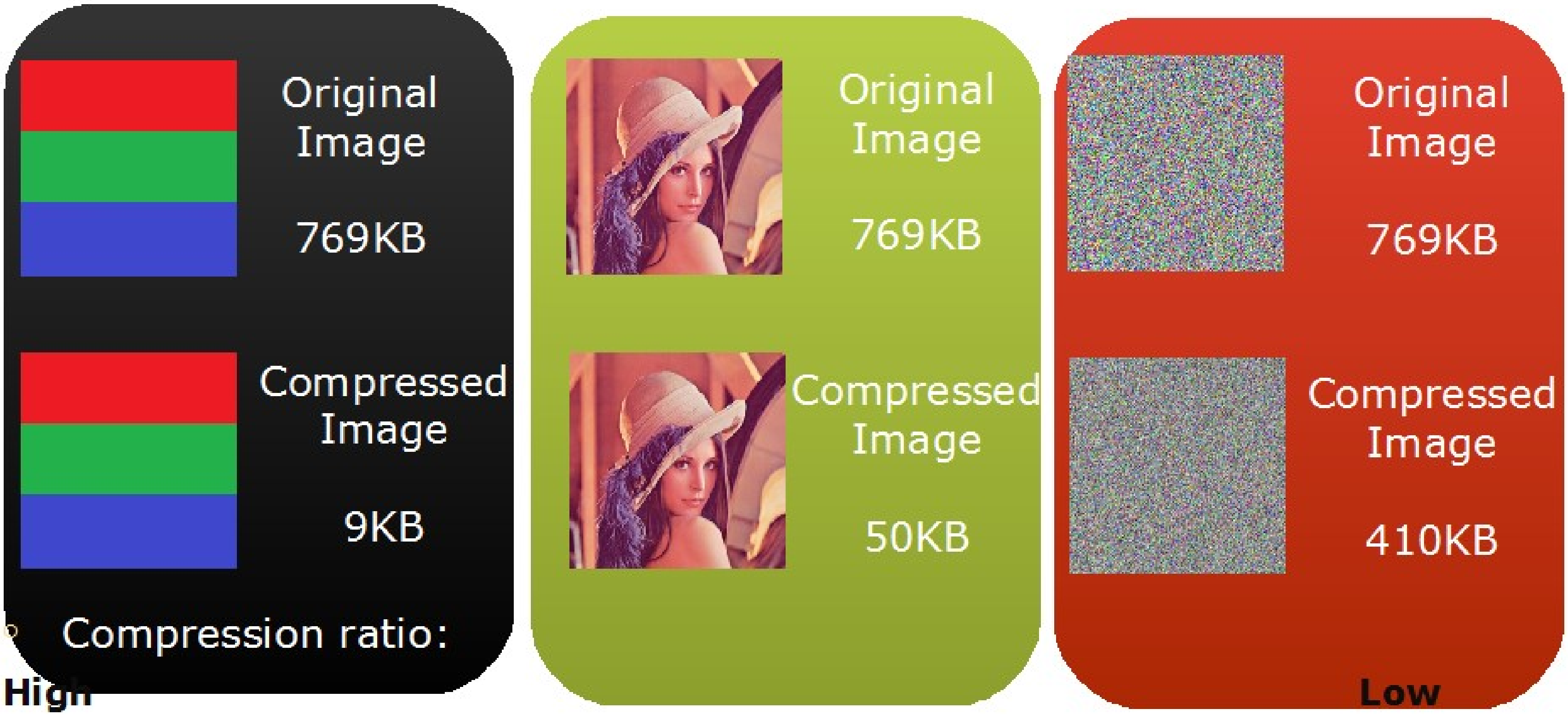
- JPEG stands for Joint Photographic Experts Group
- JPEG compression is used with .jpg and can be embedded in .tiff and .eps files.
- Used on 24-bit color files.
- Works well on photographic images.
  - Although it is a lossy compression technique, it yields an excellent quality image with high compression rates. Yields acceptable compression in the 10:1 range

JPEG Compression

- It defines three different coding systems:
  - 1. a lossy baseline coding system, adequate for most compression applications
  - 2. an extended coding system for greater compression, higher precision, or progressive reconstruction applications
  - 3. A lossless independent coding system for reversible compression

Correlation:

High **Low**



High **Low**

Compression ratio:

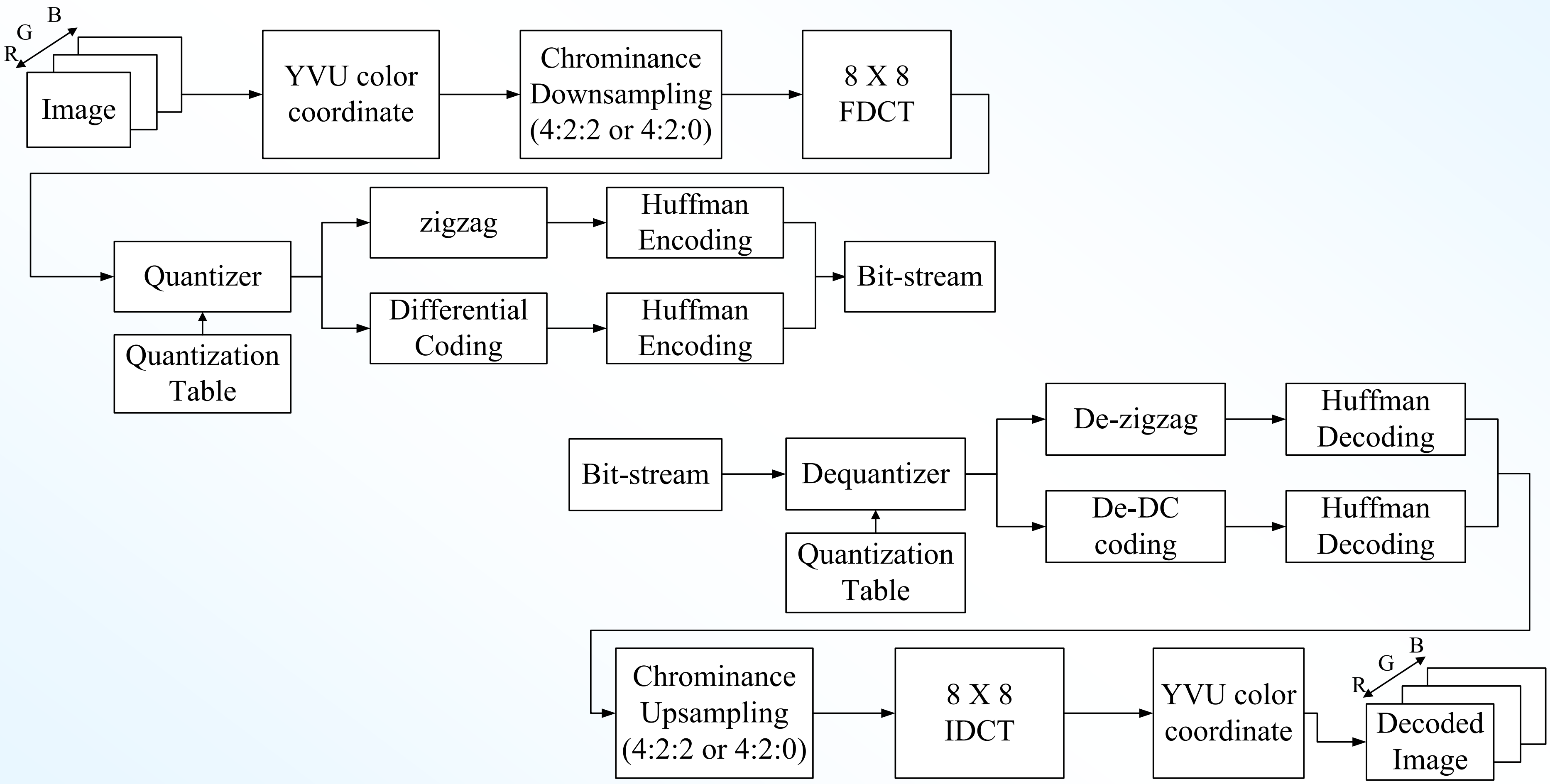
Why is JPEG is effective?

- Image data usually changes slowly across an image, especially within an 8x8 block
  - Therefore images contain much redundancy
- Experiments indicate that humans are not very sensitive to the high frequency data images
  - Therefore we can remove much of this data using transform coding
- Humans are much more sensitive to brightness (luminance) information than to color (chrominance)
  - JPEG uses chroma subsampling (4:2:0)

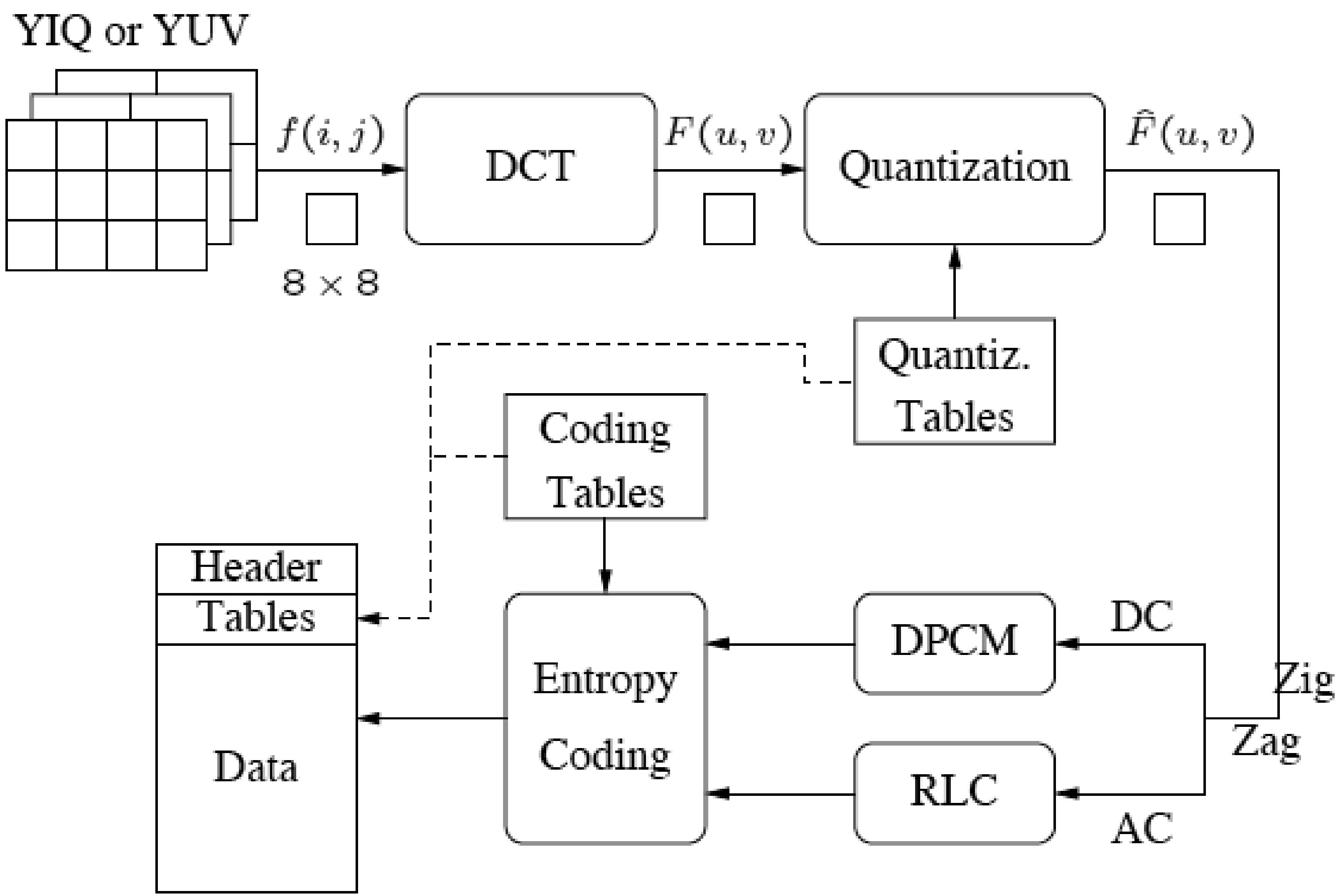
JPEG Compression

- Exploiting Psycho-visual Redundancy
- Exploit variable sensitivity of humans to colors:
  - We're more sensitive to differences between dark intensities than bright ones.
    - Encode log(intensity) instead of intensity.
  - We're more sensitive to high spatial frequencies of green than red or blue.
    - Sample green at highest spatial frequency, blue at lowest.
  - We're more sensitive to differences of intensity in green than red or blue.
    - Use variable quantization: devote most bits to green, fewest to blue.

JPEG Compression



JPEG Encoding Overview



Steps in JPEG Compression

1. (Optionally) If the color is represented in RGB mode, translate it to YUV.
2. Divide the file into 8 X 8 blocks.
3. Transform the pixel information from the spatial domain to the frequency domain with the Discrete Cosine Transform.
4. Quantize the resulting values by dividing each coefficient by an integer value and rounding off to the nearest integer.
5. Look at the resulting coefficients in a zigzag order. Do a run-length encoding of the coefficients ordered in this manner, followed by Huffman coding.

Step 1a: 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:

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

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

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

JPEG Compression

- **Exploiting Psychovisual Redundancy**

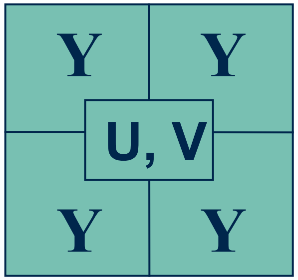
- **NTSC Video**

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} .30 & .59 & .11 \\ .60 & -.28 & -.32 \\ .21 & -.52 & .31 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

  - **Y bandlimited to 4.2 MHz**
  - **I to 1.6 MHz**
  - **Q to .6 MHz**

Step 1b: Downsampling

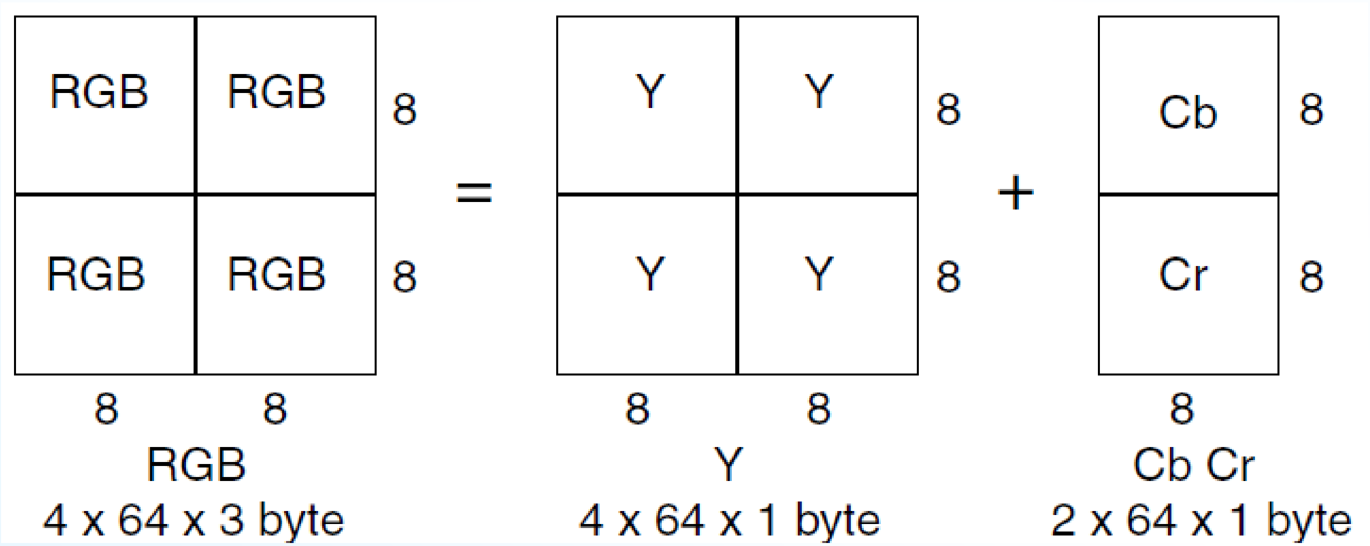
- The chrominance information can (optionally) be downsampled.
- The notation 4:1:1 means that for each block of four pixels, you have 4 samples of luminance information (Y), and 1 each of the two chrominance components (U and V).
- MCU – minimum coded unit



JPEG Compression

- **Exploiting Psychovisual Redundancy**

- **In JPEG and MPEG**
  - **Cb and Cr are sub-sampled**



Step 2: Divide into 8 x 8 blocks

- The image is divided up into 8x8 blocks
  - 2D DCT is performed on each block
  - The DCT is performed independently for each block
- !!! When a high degree of compression is requested, JPEG gives a “blocky” image result
- If the file doesn’t divide evenly into 8 X 8 blocks, extra pixels are added to the end and discarded after the compression.
- The values are shifted “left” by subtracting 128.

Discrete Cosine Transform

- The DCT transforms the data from the spatial domain to the frequency domain.
- The spatial domain shows the amplitude of the color as you move through space
- The frequency domain shows how quickly the amplitude of the color is changing from one pixel to the next in an image file.

Step 3: DCT

- The frequency domain is a better representation for the data because it makes it possible for you to separate out – and throw away – information that isn't very important to human perception.
- The human eye is not very sensitive to high frequency changes – especially in photographic images, so the high frequency data can, to some extent, be discarded.
- The color amplitude information can be thought of as a wave (in two dimensions).
- You're decomposing the wave into its component frequencies.
- For the 8 x 8 matrix of color data, you're getting an 8 X 8 matrix of coefficients for the frequency components.

Discrete Cosine Transform(DCT)
Step 3: Forward DCT

- **2D-DCT**

$$X(u,v)=\frac{4C(u)C(v)}{N^2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)\cos\left(\frac{(2m+1)u\pi}{2N}\right)\cos\left(\frac{(2n+1)v\pi}{2N}\right)$$

- **Inverse 2D-DCT**

$$x(m,n)=\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}C(u)C(v)X(u,v)\cos\left(\frac{(2m+1)u\pi}{2N}\right)\cos\left(\frac{(2n+1)v\pi}{2N}\right)$$

where

$$C(u)=\begin{cases}\dfrac{1}{\sqrt{2}} & u=0\\ 1 & u=1,\ldots,N-1\end{cases}$$

e Compression

$$\begin{bmatrix} 55 & 36 & -34 & 25 & 66 & 35 & 4 & 37 \\ 55 & 25 & -12 & 48 & 59 & 38 & 2 & 41 \\ 51 & 40 & 43 & 54 & 51 & 42 & 3 & 39 \\ 49 & 49 & 51 & 49 & 51 & 37 & 3 & 39 \\ 50 & 50 & 51 & 48 & 54 & 36 & 2 & 43 \\ 51 & 52 & 52 & 51 & 55 & 41 & 4 & 41 \\ 51 & 51 & 52 & 54 & 55 & 42 & 1 & 45 \\ 52 & 51 & 53 & 51 & 53 & 42 & 2 & 41 \end{bmatrix}$$

image block

2D-DCT

DC component

low frequency → **high frequency**

low frequency ↓ **high frequency**

$$\begin{bmatrix} 313 & 56 & -27 & 18 & 78 & -60 & 27 & -27 \\ -38 & -27 & 13 & 44 & 32 & -1 & -24 & -10 \\ -20 & -17 & 10 & 33 & 21 & -6 & -16 & -9 \\ -10 & -8 & 9 & 17 & 9 & -10 & -13 & 1 \\ -6 & 1 & 6 & 4 & -3 & -7 & -5 & 5 \\ 2 & 3 & 0 & -3 & -7 & -4 & 0 & 3 \\ 4 & 4 & -1 & -2 & -9 & 0 & 2 & 4 \\ 3 & 1 & 0 & -4 & -2 & -1 & 3 & 1 \end{bmatrix}$$

DCT block

Step 4: Quantize the Coefficients Computed by the DCT

- The DCT is lossless in that the reverse DCT will give you back exactly your initial information (ignoring the rounding error that results from using floating point numbers.)
- The values from the DCT are initially floating-point.
- They are changed to integers by quantization.
- Quantization involves dividing each coefficient by an integer between 1 and 255 and rounding off.
- The quantization table is chosen to reduce the precision of each coefficient.
- The quantization table is carried along with the compressed file.

JPEG Compression

- **Why quantization?**
  - To achieve further compression by representing DCT coefficients with no greater precision than is necessary to achieve the desired image quality
  - Generally, the "high frequency coefficients" has larger quantization values
  - Quantization makes most coefficients to be zero, it makes the compression system efficient, but it's the main source that make the system "lossy"

JPEG Compression

- Quantization is the step where we actually throw away data.
- Luminance and Chrominance Quantization Table
- Smaller numbers in the upper left direction
- larger numbers in the lower right direction
- The performance is close to the optimal condition

Quantization

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

Dequantization

$$F(u,v)_{deQ} = F(u,v)_{Quantization} \times Q(u,v)$$

JPEG Compression

Quantization

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

Dequantization

$$F(u,v)_{deQ} = F(u,v)_{Quantization} \times Q(u,v)$$

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

$$Q_C = \begin{pmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{pmatrix}$$

Quantization

- **Quantization in JPEG aims at reducing the total number of bits in the compressed image**
  - **Divide each entry in the frequency space block by an integer, then round**
  - **Use a quantization matrix Q(u, v)**

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

Quantization

- Use larger entries in Q for the higher spatial frequencies
  - These are entries to the lower right part of the matrix
  - The following slide shows the default Q(u, v) values for luminance and chrominance
    - Based on psychophysical studies intended to maximize compression ratios while minimizing perceptual distortion
    - Since after division the entries are smaller, we can use fewer bits to encode them

Quantization

Table 9.1 The 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

Table 9.2 The 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

JPEG Compression

- **The Baseline System – Quantization**

$$X'(u,v) = Round\left(\frac{X(u,v)}{Q(u,v)}\right)$$

X(u,v): original DCT coefficient
X'(u,v): DCT coefficient after quantization
Q(u,v): quantization value

Original and DCT coded block

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

$F(u, v)$

Fig. 9.2: JPEG compression for a smooth image block.

Quantized and Reconstructed Blocks

$$\begin{matrix} 32 & 6 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

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

$$\begin{matrix} 512 & 66 & -10 & 0 & 0 & 0 & 0 & 0 \\ -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -14 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ -14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

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

After IDCT and Difference from Original

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

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

Same steps on a less homogeneous block

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

70	70	100	70	87	87	150	187
85	100	96	79	87	154	87	113
100	85	116	79	70	87	86	196
136	69	87	200	79	71	117	96
161	70	87	200	103	71	96	113
161	123	147	133	113	113	85	161
146	147	175	100	103	103	163	187
156	146	189	70	113	161	163	197

$f(i,j)$

-80	-40	89	-73	44	32	53	-3
-135	-59	-26	6	14	-3	-13	-28
47	-76	66	-3	-108	-78	33	59
-2	10	-18	0	33	11	-21	1
-1	-9	-22	8	32	65	-36	-1
5	-20	28	-46	3	24	-30	24
6	-20	37	-28	12	-35	33	17
-5	-23	33	-30	17	-5	-4	20

$F(u,v)$

Steps 2 and 3

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

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

-80	-44	90	-80	48	40	51	0
-132	-60	-28	0	26	0	0	-55
42	-78	64	0	-120	-57	0	56
0	17	-22	0	51	0	0	0
0	0	-37	0	0	109	0	0
0	-35	55	-64	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

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

IDCT and Difference

70	60	106	94	62	103	146	176
85	101	85	75	102	127	93	144
98	99	92	102	74	98	89	167
132	53	111	180	55	70	106	145
173	57	114	207	111	89	84	90
164	123	131	135	133	92	85	162
141	159	169	73	106	101	149	224
150	141	195	79	107	147	210	153

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

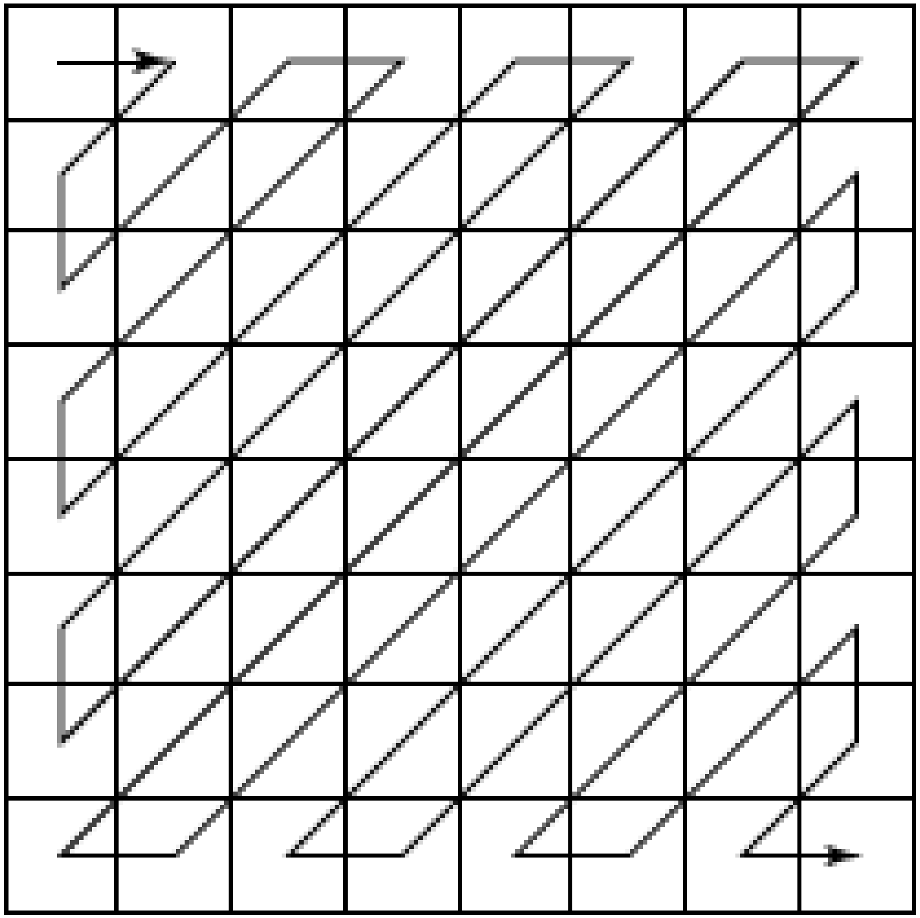
0	10	-6	-24	25	-16	4	11
0	-1	11	4	-15	27	-6	-31
2	-14	24	-23	-4	-11	-3	29
4	16	-24	20	24	1	11	-49
-12	13	-27	-7	-8	-18	12	23
-3	0	16	-2	-20	21	0	-1
5	-12	6	27	-3	2	14	-37
6	5	-6	-9	6	14	-47	44

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

Step 5: Arrange in “zigzag” order

- This is done so that the coefficients are in order of increasing frequency.
- The higher frequency coefficients are more likely to be 0 after quantization.
- This improves the compression of run-length encoding.
- Do run-length encoding and Huffman coding.

Zigzag Scan in JPEG



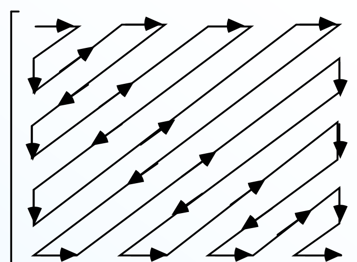
JPEG Compression

8 x8x blocks

$$\begin{bmatrix} 183 & 160 & 94 & 153 & 194 & 163 & 132 & 16 \\ 183 & 153 & 116 & 176 & 187 & 166 & 130 & 16 \\ 179 & 168 & 171 & 182 & 179 & 170 & 131 & 16 \\ 177 & 177 & 179 & 177 & 179 & 165 & 131 & 16 \\ 178 & 178 & 179 & 176 & 182 & 164 & 130 & 17 \\ 179 & 180 & 180 & 179 & 183 & 169 & 132 & 16 \\ 179 & 179 & 180 & 182 & 183 & 170 & 129 & 17 \\ 180 & 179 & 181 & 179 & 181 & 170 & 130 & 16 \end{bmatrix} \xrightarrow{-128} \begin{bmatrix} 55 & 36 & -34 & 25 & 66 & 35 & 4 & 37 \\ 55 & 25 & -12 & 48 & 59 & 38 & 2 & 41 \\ 51 & 40 & 43 & 54 & 51 & 42 & 3 & 39 \\ 49 & 49 & 51 & 49 & 51 & 37 & 3 & 39 \\ 50 & 50 & 51 & 48 & 54 & 36 & 2 & 43 \\ 51 & 52 & 52 & 51 & 55 & 41 & 4 & 41 \\ 51 & 51 & 52 & 54 & 55 & 42 & 1 & 45 \\ 52 & 51 & 53 & 51 & 53 & 42 & 2 & 41 \end{bmatrix}$$

DCT

$$\begin{bmatrix} 313 & 56 & -27 & 18 & 78 & -60 & 27 & -27 \\ -38 & -27 & 13 & 44 & 32 & -1 & -24 & -10 \\ -20 & -17 & 10 & 33 & 21 & -6 & -16 & -9 \\ -10 & -8 & 9 & 17 & 9 & -10 & -13 & 1 \\ -6 & 1 & 6 & 4 & -3 & -7 & -5 & 5 \\ 2 & 3 & 0 & -3 & -7 & -4 & 0 & 3 \\ 4 & 4 & -1 & -2 & -9 & 0 & 2 & 4 \\ 3 & 1 & 0 & -4 & -2 & -1 & 3 & 1 \end{bmatrix} \xrightarrow{\text{scalar quantization}} \begin{bmatrix} 20 & 5 & -3 & 1 & 3 & -2 & 1 & 0 \\ -3 & -2 & 1 & 2 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



zig-zag scan

Run-Length Coding

- Now the RLC step replaces values in a 64-vector (previously an 8x8 block) by a pair (RUNLENGTH, VALUE), where RUNLENGTH is the number of zeroes in the run and VALUE is the next non-zero value
  - From the first example we have (32, 6, -1, -1, 0, -1, 0, 0, 0, -1, 0, 0, 1, 0, 0, …, 0)
  - This becomes (0,6) (0,-1)(1,-1)(3,-1)(2,1)(0,0) - Note that DC coefficient is ignored

Coding of DC Coefficients

- Now we handle the DC coefficients
  - 1 DC per block
  - DC coefficients may vary greatly over the whole image, but slowly from one block to its neighbor (once again, zigzag order)
  - So apply Differential Pulse Code Modulation (DPCM) for the DC coefficients
  - If the first five DC coefficients are 150, 155, 149, 152, 144, we come up with DPCM code- 150, 5, -6, 3, -8

Entropy Coding

- Now we apply entropy coding to the RLC coded AC coefficients and the DPCM coded DC coefficients
  - The baseline entropy coding method uses Huffman coding on images with 8-bit components
  - DPCM-coded DC coefficients are represented by a pair of symbols (SIZE, AMPLITUDE)
    - SIZE = number of bits to represent coefficient
    - AMPLITUDE = the actual bits

Entropy Coding

- **The size category for the different possible amplitudes is shown below**
  - **DPCM values might require more than 8 bits and might be negative**

SIZE	AMPLITUDE
1	-1, 1
2	-3, -2, 2, 3
3	-7..-4, 4..7
4	-15..-8, 8..15
.	.
.	.
.	.
10	-1023..-512, 512..1023

Entropy Coding

– One’s complement is used for negative numbers
– Codes 150, 5, -6, 3, -8 become
– (8, 10010110), (3, 101), (2, 11), (4, 0111)
– Now the SIZE is Huffman coded
  • Expect lots of small SIZEs
– AMPLITUDE is not Huffman coded
  • Pretty uniform distribution expected, so probably not worth while

Huffman Coding for AC Coefficients

- AC coefficients have been RL coded and represented by symbol pairs (RUNLENGTH, VALUE)
  - VALUE is really a (SIZE, AMPLITUDE) pair
    - RUNLENGTH and SIZE are each 4-bit values stored in a single byte - Symbol1
      - For runs greater than 15, special code (15, 0) is used
    - Symbol2 is the AMPLITUDE
    - Symbol1 is run-length coded, Symbol 2 is not

EXAMPLE 8.17:
JPEG baseline coding and decoding.

52	55	61	66	70	61	64	73
63	59	66	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	63	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

EXAMPLE 8.17:
JPEG baseline coding and decoding.

−76	−73	−67	−62	−58	−67	−64	−55
−65	−69	−62	−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	−65	−70	−53
−43	−57	−64	−69	−73	−67	−63	−45
−41	−49	−59	−60	−63	−52	−50	−34

EXAMPLE 8.17:
JPEG baseline coding and decoding.

−415	−29	−62	25	55	−20	−1	3
7	−21	−62	9	11	−7	−6	6
−46	8	77	−25	−30	10	7	−5
−50	13	35	−15	−9	6	0	3
11	−8	−13	−2	−1	1	−4	1
−10	1	3	−3	−1	0	2	−1
−4	−1	2	−1	2	−3	1	−2
−1	−1	−1	−2	−1	−1	0	−1

EXAMPLE 8.17:
JPEG baseline coding and decoding.

−26	−3	−6	2	2	0	0	0
1	−2	−4	0	0	0	0	0
−3	1	5	−1	−1	0	0	0
−4	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

EXAMPLE 8.17:
JPEG baseline coding and decoding.

[−26 −3 1 −3 −2 −6 2 −4 1 −4 1 1 5 0 2 0 0 −1 2 0 0 0 0 0 −1 −1 EOB]

The construction of the default JPEG code for the reordered coefficient sequence begins with the computation of the difference between the current DC coefficient and that of the previously encoded sub-image.

Assuming the DC coefficient of the transformed and quantized sub-image to its immediate left was -17.

The resulting of DPCM difference is [-26 – (-17)] = -9.

1010110 0100 001 0100 0101 100001 0110 100011 001 100011 001
001 100101 11100110 110110 0110 11110100 000 1010

EXAMPLE 8.17: JPEG baseline coding and decoding.

[−26 −3 1 −3 −2 −6 2 −4 1 −4 1 1 5 0 2 0 0 −1 2 0 0 0 0 0 −1 −1 EOB]

Each default AC Huffman code word depends on the number of zero-valued coefficients preceding the nonzero coefficient to be coded, as well as the magnitude category of the nonzero coefficient.

1010110 0100 001 0100 0101 100001 0110 100011 001 100011 001
001 100101 11100110 110110 0110 11110100 000 1010

EXAMPLE 8.17:
JPEG baseline coding and decoding.

−26	−3	−6	2	2	0	0	0
1	−2	−4	0	0	0	0	0
−3	1	5	−1	−1	0	0	0
−4	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

EXAMPLE 8.17: JPEG baseline coding and decoding.

−416	−33	−60	32	48	0	0	0
12	−24	−56	0	0	0	0	0
−42	13	80	−24	−40	0	0	0
−56	17	44	−29	0	0	0	0
18	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

EXAMPLE 8.17:
JPEG baseline coding and decoding.

−70	−64	−61	−64	−69	−66	−58	−50
−72	−73	−61	−39	−30	−40	−54	−59
−68	−78	−58	−9	13	−12	−48	−64
−59	−77	−57	0	22	−13	−51	−60
−54	−75	−64	−23	−13	−44	−63	−56
−52	−71	−72	−54	−54	−71	−71	−54
−45	−59	−70	−68	−67	−67	−61	−50
−35	−47	−61	−66	−60	−48	−44	−44

EXAMPLE 8.17:
JPEG baseline
coding and
decoding.

58	64	67	64	59	62	70	78
56	55	67	89	98	88	74	69
60	50	70	119	141	116	80	64
69	51	71	128	149	115	77	68
74	53	64	105	115	84	65	72
76	57	56	74	75	57	57	74
83	69	59	60	61	61	67	78
93	81	67	62	69	80	84	84

EXAMPLE 8.17:
JPEG baseline coding and decoding.

−6	−9	−6	2	11	−1	−6	−5
7	4	−1	1	11	−3	−5	3
2	9	−2	−6	−3	−12	−14	9
−6	7	0	−4	−5	−9	−7	1
−7	8	4	−1	6	4	3	−2
3	8	4	−4	2	6	1	1
2	2	5	−1	−6	0	−2	5
−6	−2	2	6	−4	−4	−6	10

Examples of JPEG Compression

a b c
d e f

FIGURE 8.32 Two JPEG approximations of Fig. 8.9(a). Each row contains a result after compression and reconstruction, the scaled difference between the result and the original image, and a zoomed portion of the reconstructed image.

JPEG at 0.125 bpp (enlarged)

C. Christopoulos, A. Skodras, T. Ebrahimi, JPEG2000 (online tutorial)

JPEG2000 at 0.125 bpp

C. Christopoulos, A. Skodras, T. Ebrahimi, JPEG2000 (online tutorial)

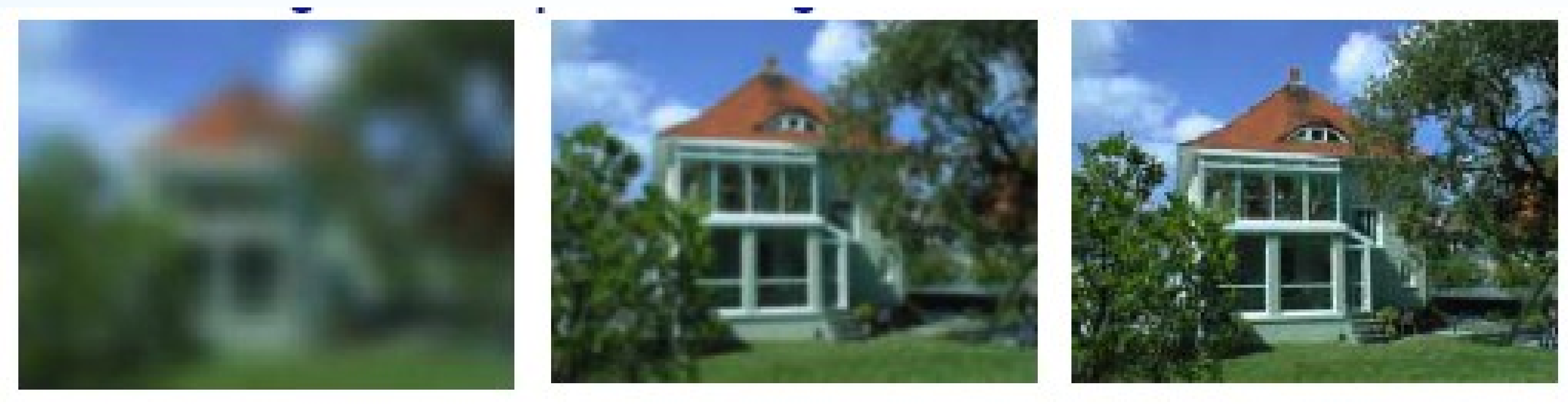
JPEG Modes

- JPEG supports several different modes
  - Sequential Mode
  - Progresssive Mode
  - Hierarchical Mode
  - Lossless Mode
- Sequential is the default mode
  - Each image component is encoded in a single left-to-right, top-to-bottom scan
    - This is the mode we have been describing

Progressive Mode

- Progressive mode delivers low-quality versions of the image quickly, and then fills in the details in successive passes
- This is useful for web browsers, where the image download might take a long time
  - The user gets an approximate image quickly
  - Can be done by sending the DC coefficient and a few AC coefficients first
  - Next send some more (low spatial resolution) AC coefficients, and continue in this way until all of the coefficients have been sent

Sequential vs. Progressive



Hierarchical Mode

- Hierarchical mode encodes the image at several different resolutions
- These resolutions can be transmitted in multiple passes with increased resolution at each pass
- The process is described in the following slides

Hierarchical Mode

Encoder for a Three-level Hierarchical JPEG

1. Reduction of image resolution:

   Reduce resolution of the input image f (e.g., 512×512) by a factor of 2 in each dimension to obtain f_2 (e.g., 256×256). Repeat this to obtain f_4 (e.g., 128×128).

2. Compress low-resolution image f_4:

   Encode f_4 using any other JPEG method (e.g., Sequential, Progressive) to obtain F_4.

3. Compress difference image d_2:

   (a) Decode F_4 to obtain $\tilde{f}_4$. Use any interpolation method to expand $\tilde{f}_4$ to be of the same resolution as f_2 and call it $E(\tilde{f}_4)$.

   (b) Encode difference $d_2 = f_2 - E(\tilde{f}_4)$ using any other JPEG method (e.g., Sequential, Progressive) to generate D_2.

4. Compress difference image d_1:

   (a) Decode D_2 to obtain $\tilde{d}_2$; add it to $E(\tilde{f}_4)$ to get $\tilde{f}_2 = E(\tilde{f}_4) + \tilde{d}_2$ which is a version of f_2 after compression and decompression.

   (b) Encode difference $d_1 = f - E(\tilde{f}_2)$ using any other JPEG method (e.g., Sequential, Progressive) to generate D_1.

Hierarchical Mode

Decoder for a Three-level Hierarchical JPEG

1. Decompress the encoded low-resolution image F_4:
   - Decode F_4 using the same JPEG method as in the encoder to obtain $\tilde{f}_4$.
2. Restore image $\tilde{f}_2$ at the intermediate resolution:
   - Use $E(\tilde{f}_4) + \tilde{d}_2$ to obtain $\tilde{f}_2$.
3. Restore image $\tilde{f}$ at the original resolution:
   - Use $E(\tilde{f}_2) + \tilde{d}_1$ to obtain $\tilde{f}$.

Hierarchical Mode

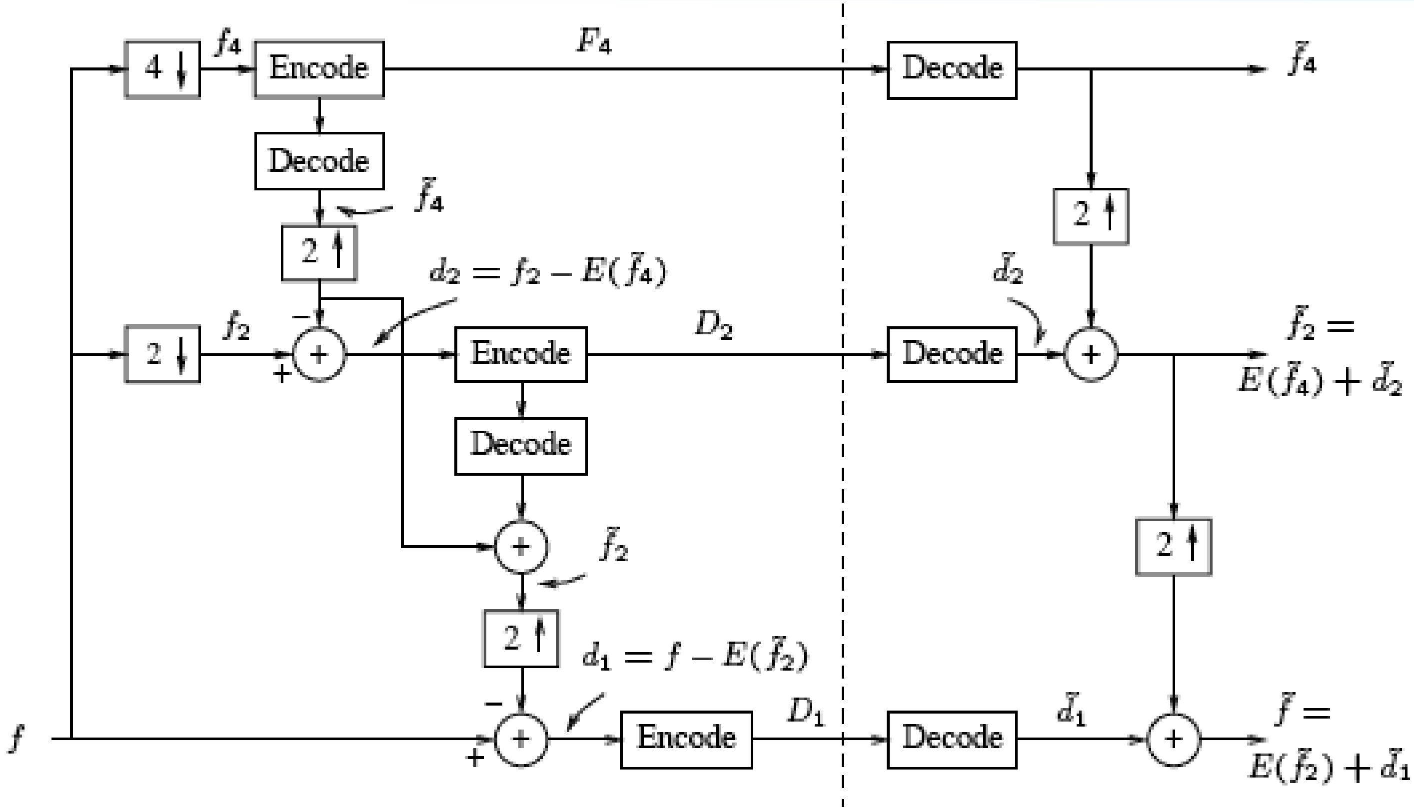
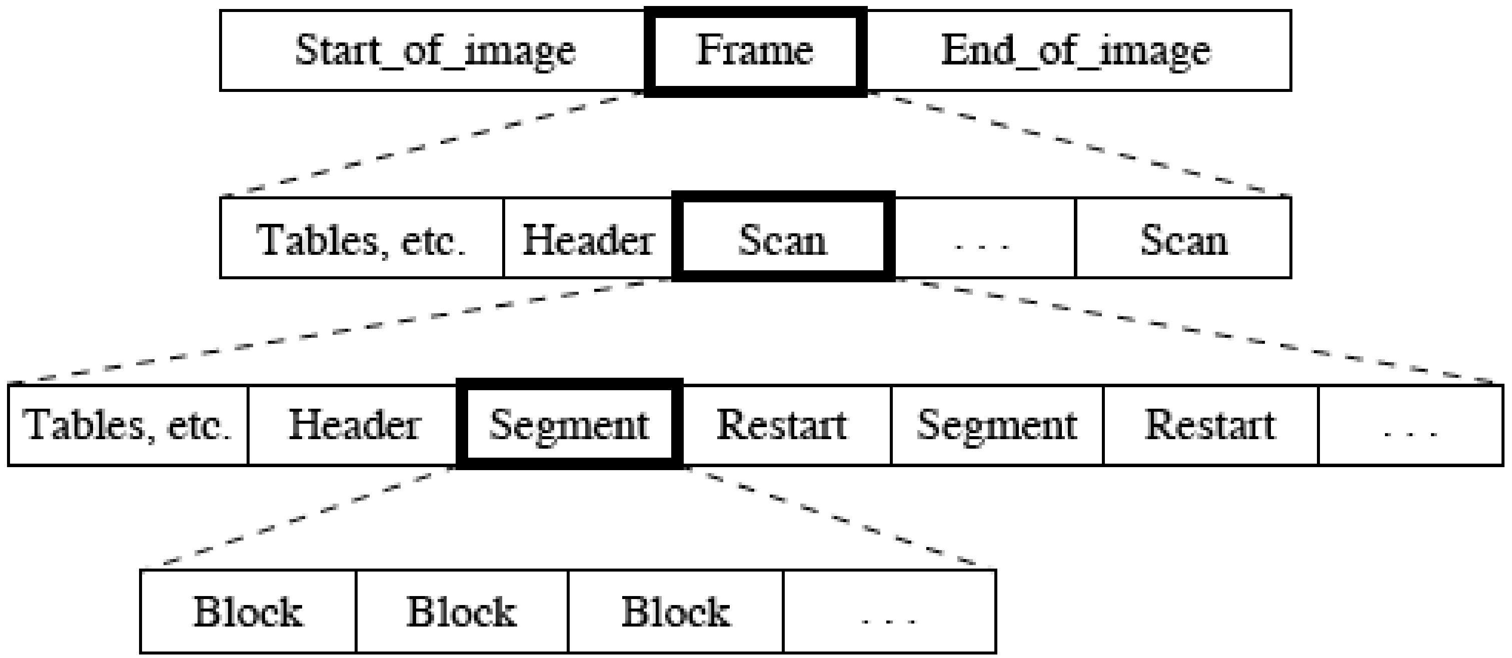


Fig. 9.5: Block diagram for Hierarchical JPEG.

JPEG Bitstream

- The JPEG hierarchical organization is described in the next slide
  - Frame is a picture
  - Scan is a picture component
  - Segment is a group of blocks
  - Frame header inlcudes
    - Bits per pixel
    - Size of image
    - Quantization table etc.
  - Scan header includes
    - Number of components
    - Huffman coding tables, etc.

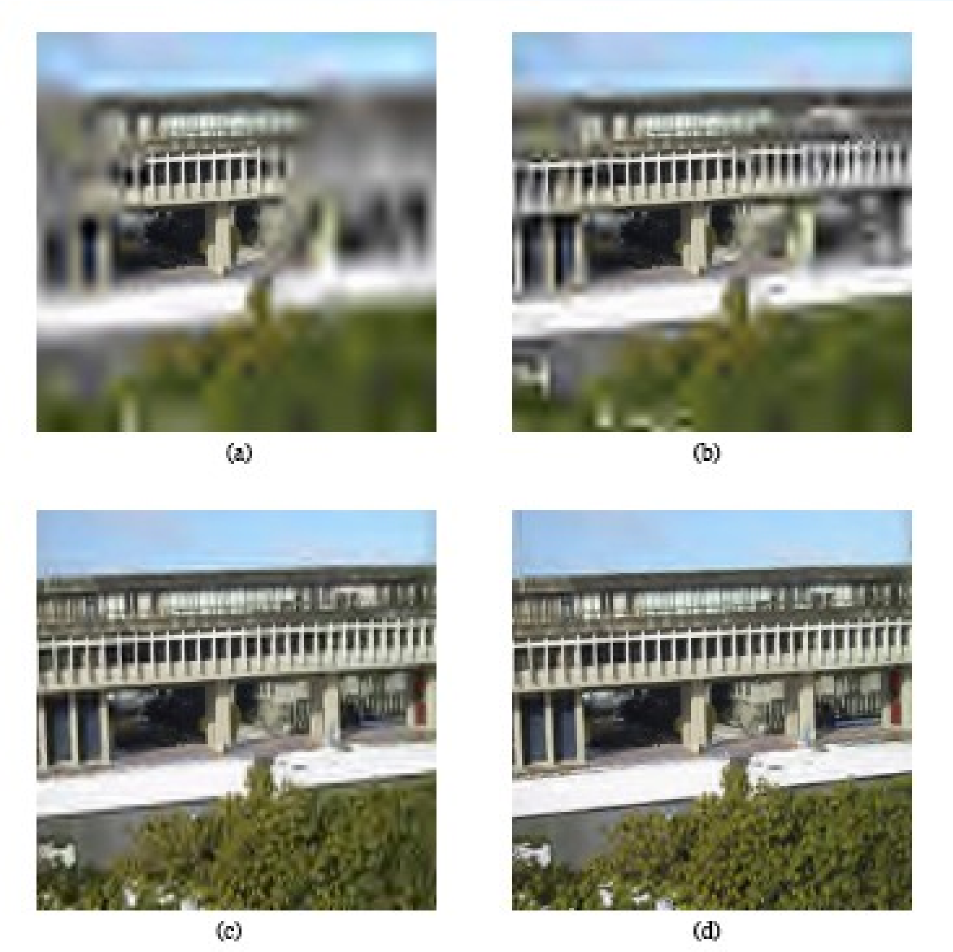
JPEG Bitstream



JPEG2000

- **JPEG2000 (extension jp2) is the latest series of standards from the JPEG committee**
  - Uses wavelet technology
  - Better compression than JPG
  - Superior lossless compression
  - Supports large images and images with many components
  - Region-of-interest coding
  - Compound documents
  - Computer-generated imagery
  - Other improvements over JPG

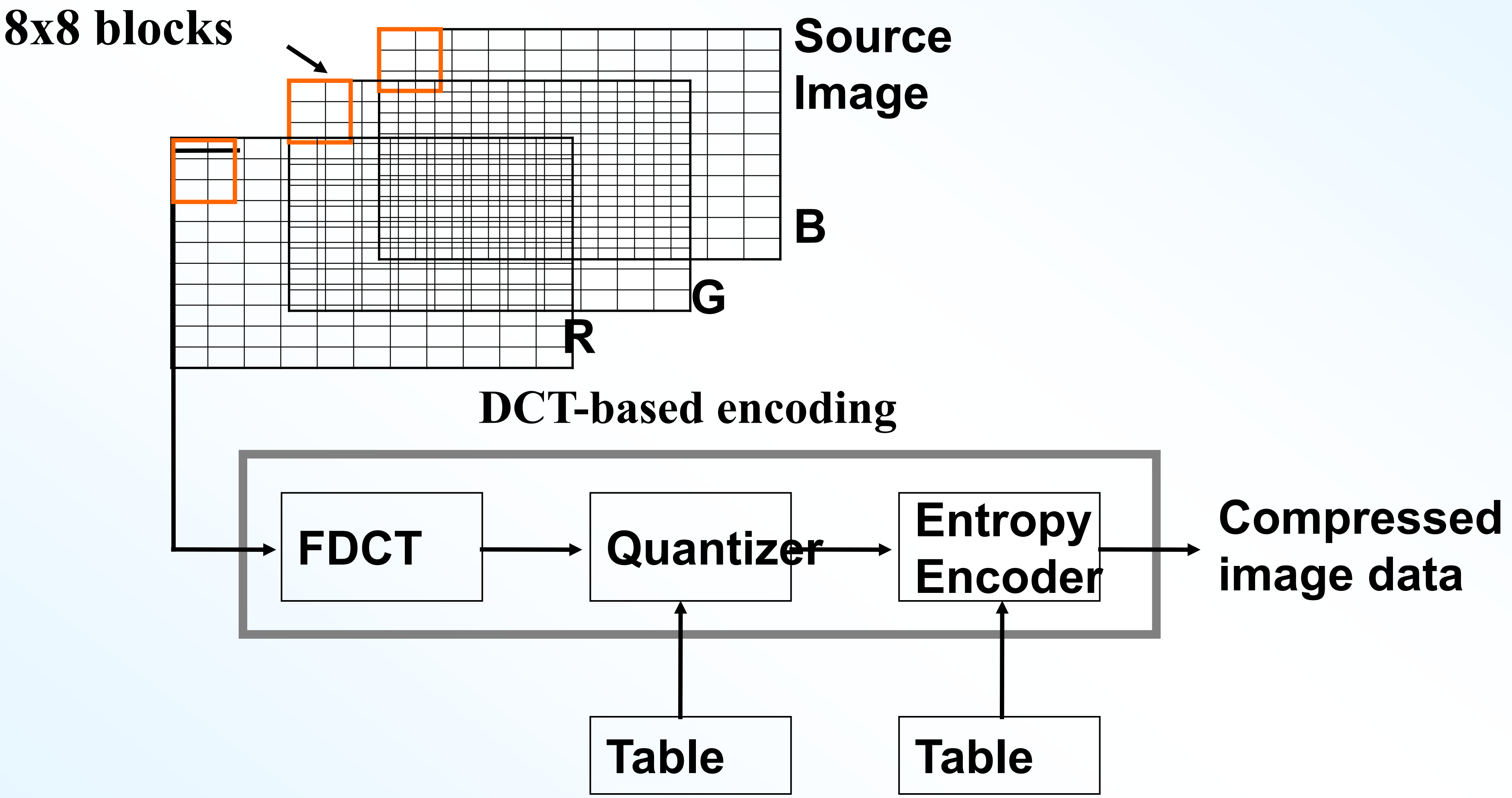
Region-of-Interest Coding



JBIG

- **JBIG (Joint Bi-Level Image Processing Group) is a standard for coding binary images**
  - Faxes, scanned documents, etc.
  - These have characteristics different from color/greyscale images which lend themselves to different coding techniques
  - JBIG - lossless coding
  - JBIG2 - both, lossless and lossy
    - Model-based coding

JPEG Compression



De facto Quantization Table

Eye becomes less sensitive

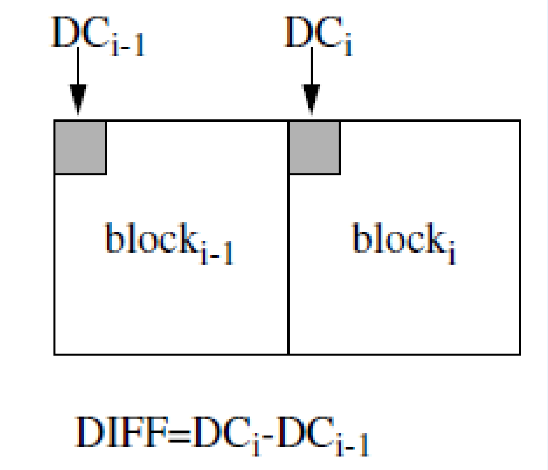
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Eye becomes less sensitive

Entropy Encoding

- Compress sequence of quantized DC and AC coefficients from quantization step
  - further increase compression, without loss
- Separate DC from AC components
  - DC components change slowly, thus will be encoded using difference encoding

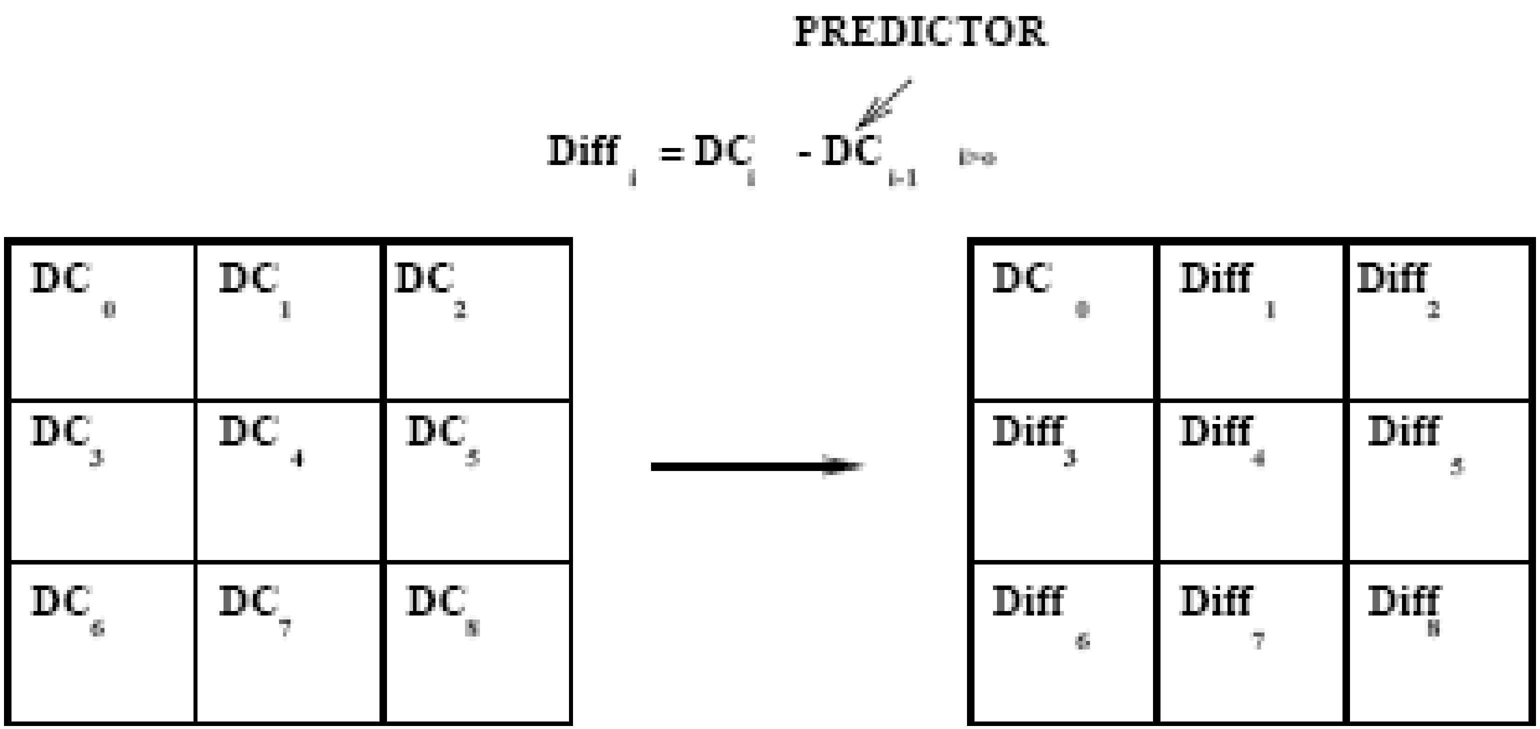
DC Encoding



- DC represents average intensity of a block
  - encode using difference encoding scheme
  - use 3x3 pattern of blocks

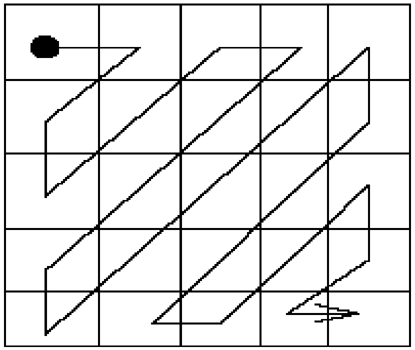
- The DC-coefficients determine the fundamental color of the units. Since this changes little between neighboring data units, the differences between successive DC-coefficients are very small values. Thus each DC-coefficient is encoded by subtracting the DC-coefficient of the previous data unit.

Difference Coding applied to DC Coefficients



AC Encoding

- Use zig-zag ordering of coefficients:
  - orders frequency components from low->high
  - produce maximal series of 0s at the end
  - Ordering helps to apply efficiently entropy encoding
- Zig-zag ordering allows better entropy encoding due to DC and AC coefficient distribution in the 8x8 block matrix.
- Apply Huffman coding
- Apply RLE on AC zero values



Huffman Encoding

- Sequence of DC difference indices and values along with RLE of AC coefficients
- Apply Huffman encoding to sequence
- Attach appropriate headers
- Finally have the JPEG image!