UNIVERSITÀ DEGLI STUDI
DI MODENA E REGGIO EMILIA

Lecture notes for Multimedia Data Processing

JPEG Compression

Last updated on: 27/03/2020

The JPEG Compression Standard

- JPEG is an abbreviation that stands for Joint Photographic Experts Group and it is a standard for the compression of static images in continuous color tones. It was designed to encode photographic images, as opposed to computer-generated synthetic images.
- It is a very common standard.
- It is not a single algorithm, but it uses different coding techniques:
  - Encoding via Discrete Cosine Transform (DCT)
  - Run Length Encoding (RLE)
  - Huffman Encoding
- Four different operating modes:
  - Lossless JPEG
  - Sequential (Baseline) JPEG
  - Progressive JPEG
  - Hierarchical JPEG

Lossless JPEG

- Exploits a linear prediction technique.
- It provides 8 different prediction schemes.
- The *residual* image is constructed as the difference between the current pixel *x* and the *y* value obtained with the prediction.
- The residual image is compressed with Huffmann

	c	b
	a	x

Prediction schemes used in Lossless JPEG

$$y = 0$$

$$y = a$$

$$y = b$$

$$y = c$$

$$y = a + b - c$$

$$y = a + \frac{b - c}{2}$$

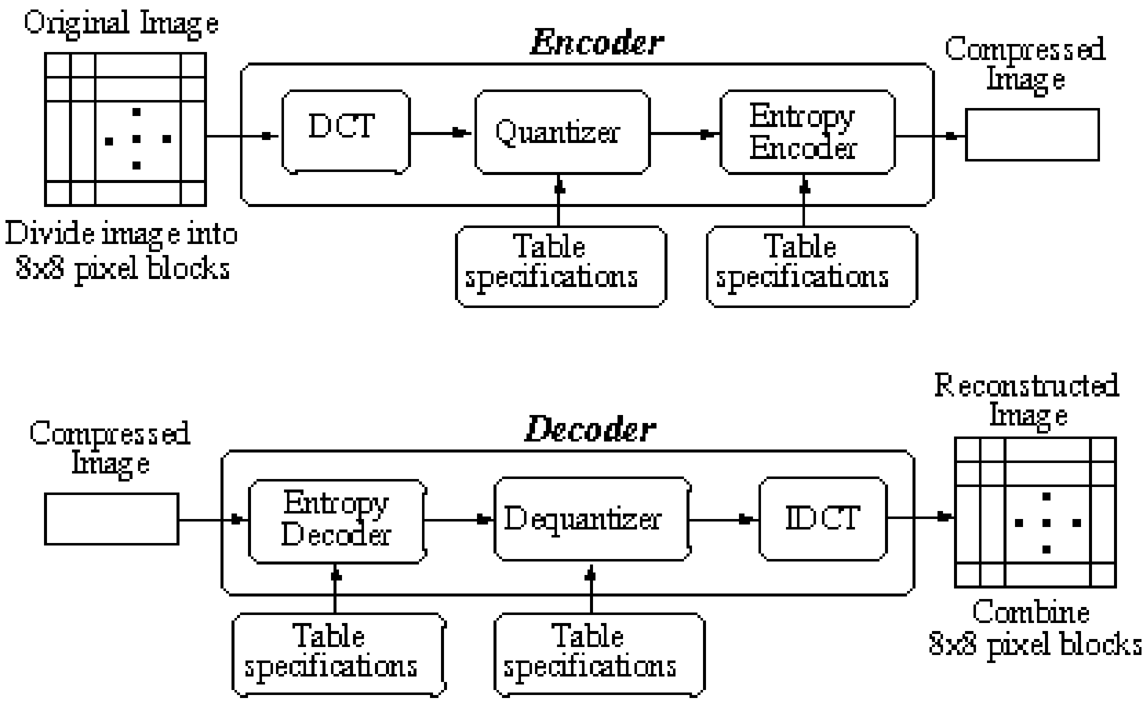
$$y = b + \frac{a - c}{2}$$

$$y = \frac{a + b}{2}$$

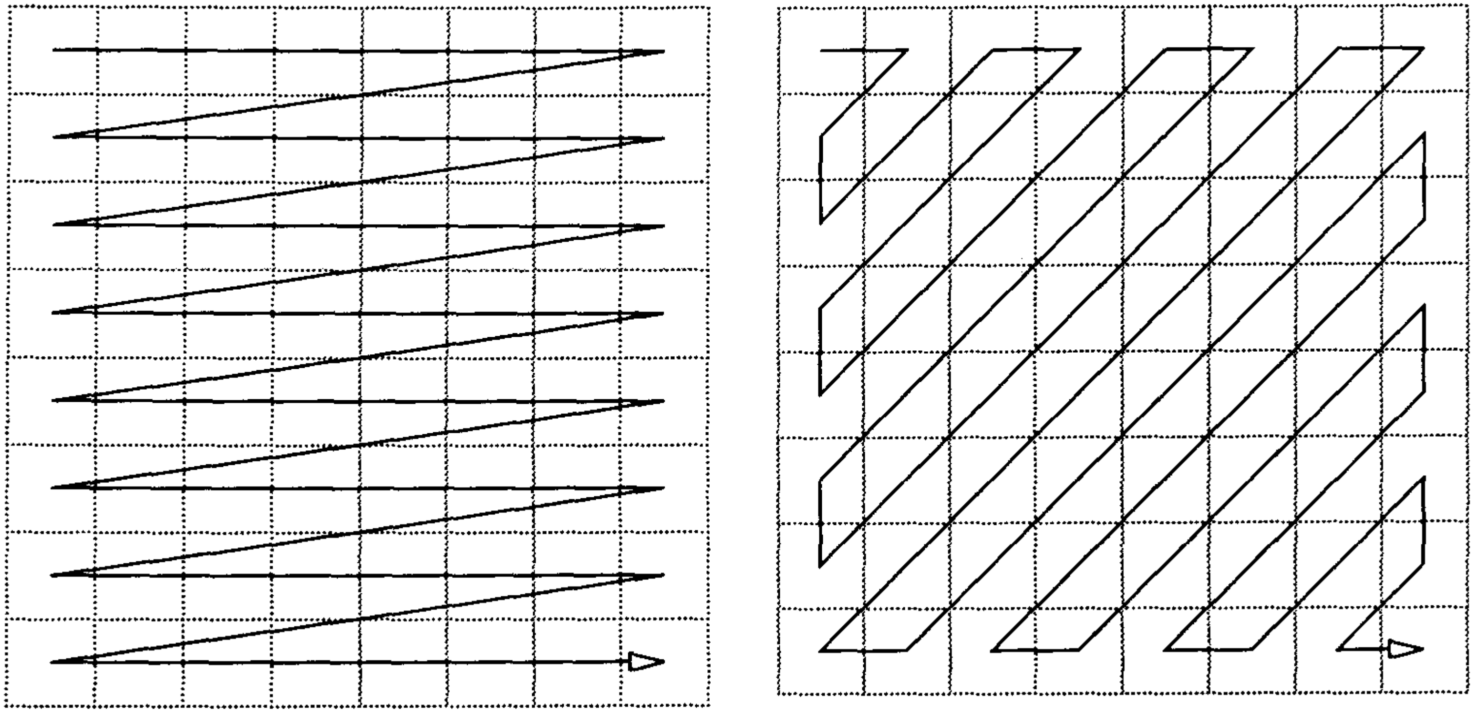
Baseline JPEG

- JPEG is a *color-blind* compression algorithm: it does not care what it is compressing. The input is an array of n-bits per pixel values. If the image is in color (for example RGB) each plane is compressed separately.
- Each pixel is scaled by subtracting 2^{n-1} from it
- The image is divided into non-overlapping 8x8 blocks.
- Each block is processed with the DCT transform obtaining 8x8 coefficients.
- The coefficients are quantized (divided) using a table specified by the standard, which provides a different quantization value for each frequency.
- The coefficients are sorted using a zig-zag path.
- The quantized and ordered coefficients are encoded using a variable bit length entropic encoder (Huffman encoding tables)

Encoding/Decoding JPEG Schema



Conventional and Zig-Zag Sorting in an 8x8 Matrix



Discrete Cosine Transform

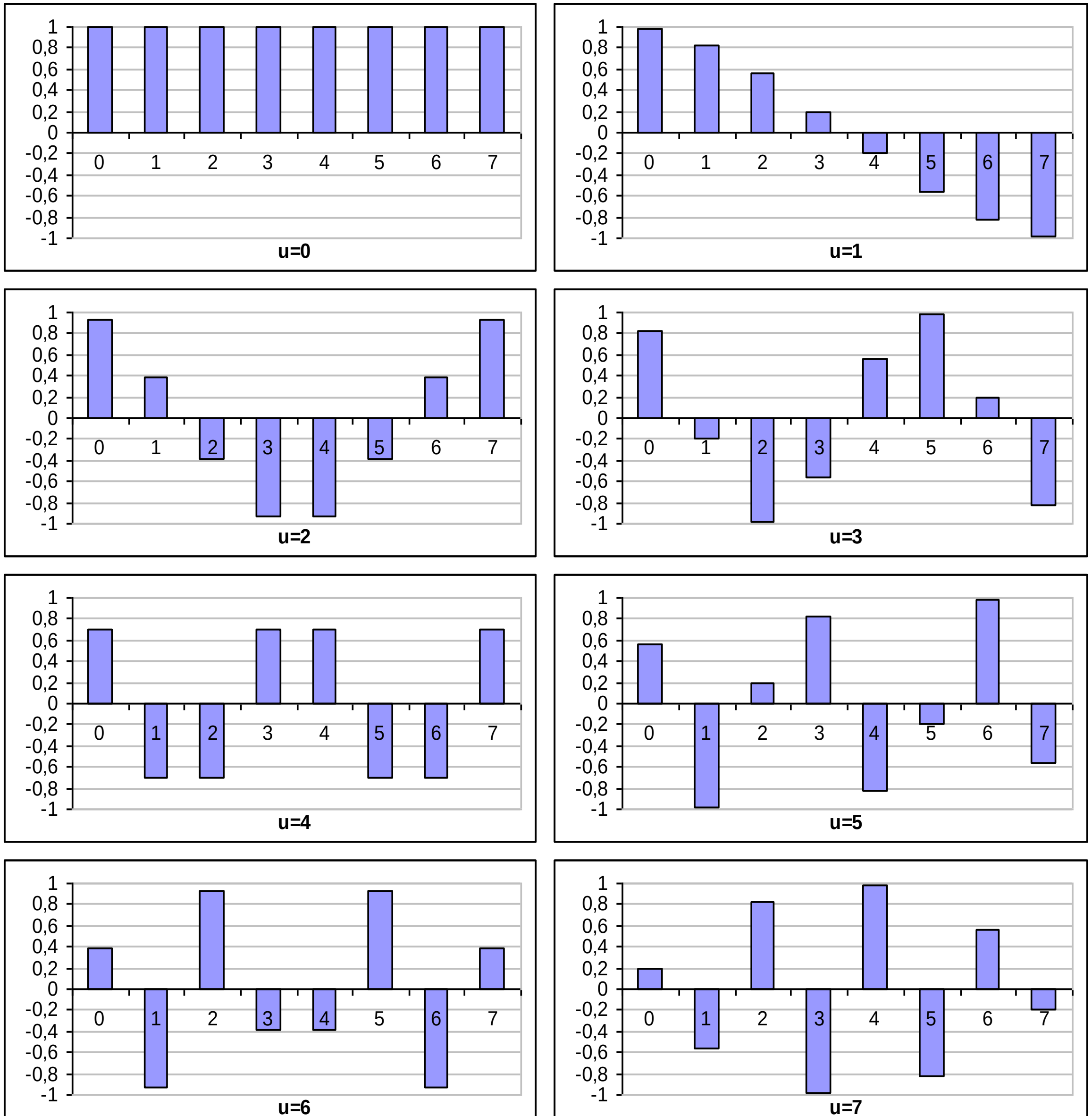
- It is a variant of the Fourier transform that provides real coefficients.
- It has interesting properties for compression, since it separates the spatial frequencies by polarizing the lower ones at lower values of the coefficients.
- The direct and inverse equations are as follows (FDCT stands for Forward DCT while IDCT stands for Inverse DCT):

$$FDCT \quad S_{uv} = \frac{1}{4} C_u C_v \sum_{y=0}^{7} \sum_{x=0}^{7} s_{xy} \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16}$$

$$IDCT \quad s_{xy} = \frac{1}{4} \sum_{v=0}^{7} \sum_{u=0}^{7} C_u C_v S_{uv} \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16}$$

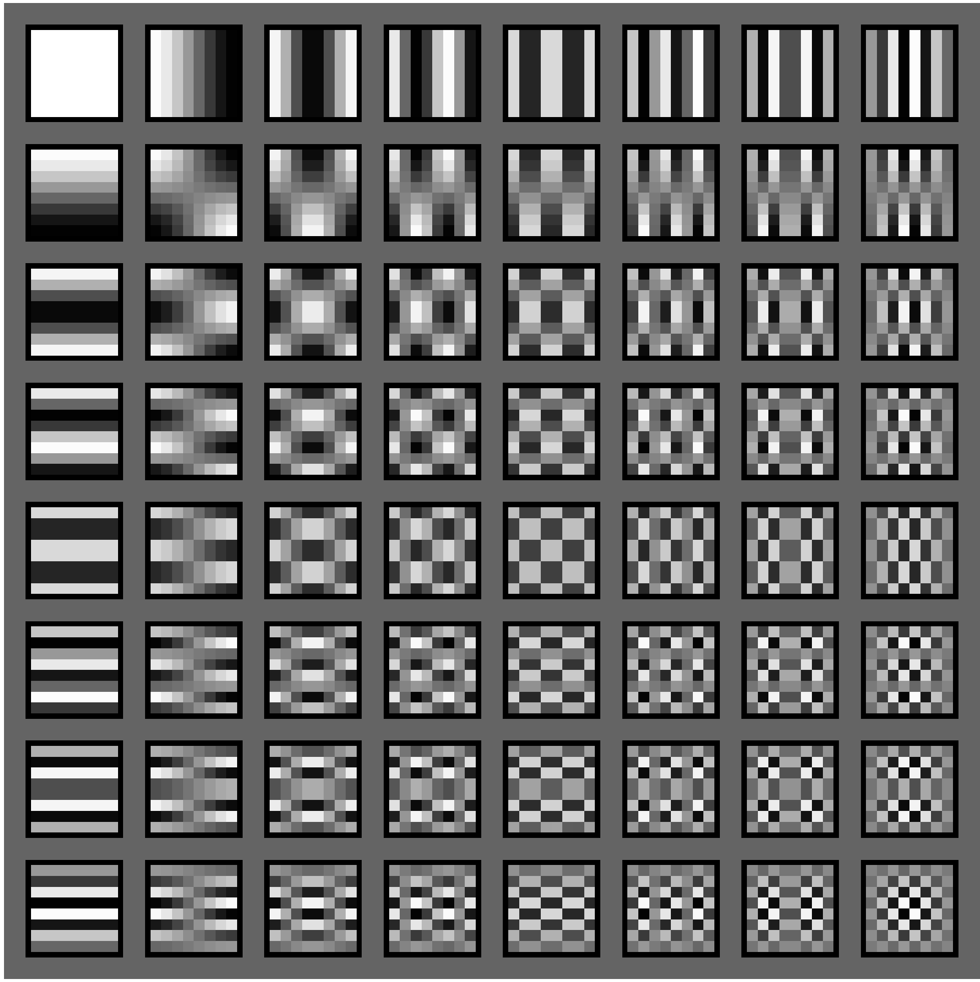
$$C_x = \begin{cases} \frac{1}{\sqrt{2}} & x = 0 \\ 1 & otherwise \end{cases}$$

Cosine Values as the Frequency Changes



DCT 2D Cosines Values

- The image on the right shows the values of the cosines, ordered as the u and v coefficients of the transform vary.
- These are the values that multiply the pixels of the original image.



Quantization Matrix

- The quantization matrix describes the precision with which each coefficient of the DCT transform will be represented.
- The matrix shown here is defined as **an example** in the JPEG standard and serves as a starting point for most compression libraries.
- The desired level or quality of compression is obtained by multiplying or dividing the coefficients of this matrix, thus increasing or decreasing its values proportionally.

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

Baseline JPEG

- DC coefficient:
  - It is the coefficient (0,0) of each 8x8 block
  - It is proportional to the average of the pixel values in the block
  - It derives from Direct Current
- AC coefficients: these are the remaining 63 coefficients in each 8x8 block
- The DC coefficient is compressed differently from the others, since the average value of a block will be similar to that of the neighboring blocks.
- The DC coefficients are then encoded by difference with respect to the previously encoded block.

Encoding of the DC Coefficient

- Working with 8-bit/pixel value maps, 11-bit precision DC coefficients are obtained.
- Differential coding will therefore require 12-bit precision
- A *Category Table* is defined for the DC values with 12 possible ranges of values.
- A 4-bit index allows you to identify from which category the current coefficient belongs to.
- SSSS: index that identifies the category (4 s to indicate 4 bits)
- A Huffman table is then defined with the SSSS variable length encoding

SSSS	**Differential DC coefficient values**
0	0
1	-1,1
2	-3,-2,2,3
3	-7...-4,4...7
4	-15...-8,8...15
5	-31...-16,16...31
6	-63...-32,32...63
7	-127...-33,33...127
8	-255...-128,128...255
9	-511...-256,256...511
10	-1023...-512,512...1023
11	-2047...-1024,1024...2047

Encoding of the DC Coefficient

- The DC coefficient for the current block is calculated.
- Subtract the previously coded value from this by creating the DC differential coefficient (we call it DIFF).
- The value of the SSSS category to which DIFF belongs is found.
- For each category, SSSS bits are added to the SSSS Huffman code to identify which DIFF was generated.
- When DIFF is positive, the least significant SSSS bits of DIFF are added.
- When DIFF is negative, the least significant SSSS bits of DIFF -1 are added.
- In practice, the first bit of those added is 0 if DIFF is negative and 1 if it is positive.

Encoding of the AC Coefficient

- As before, the coefficients are grouped into groups identified by SSSS for which a Huffman coding is available
- To each non-zero AC coefficient a class NNNNSSSS can be associated.
- NNNN represents the number of null coefficients encountered by the last coded coefficient and before the current one.
- There are 16 possible Run Lengths (0-15): NNNN and 10 categories for AC values: SSSS with two special cases:
  - 16 zeros in a row (ZRL): 11110000
  - End of block (EOB): 00000000
- Huffman table size: 16 * 10 + 2 = 162 entries.

SSSS	AC coefficient values
1	-1,1
2	-3,-2,2,3
3	-7...-4,4...7
4	15...-8,8...15
5	-31...-16,16...31
6	-63...-32,32...63
7	-127...-33,33...127
8	-255...-128,128...255
9	-511...-256,256...511
10	-1023...-512,512...1023

Encoding of the AC Coefficient

- The class NNNNSSSS is calculated for the coefficient AC (Ai)
- The Huffman code corresponding to NNNNSSSS is determined
- As with the DC coefficient, SSSS bits are added to the class Huffman code to identify which Ai we are coding for.
- When Ai is positive, the least significant SSSS bits of Ai are added. When Ai is negative, the least significant SSSS bits of Ai-1 are added.

Example

Blocco originale

95	91	91	91	84	84	84	84
109	99	95	91	91	84	84	84
109	109	99	99	91	91	91	91
120	120	109	106	99	95	91	91
131	131	120	109	106	99	95	91
145	139	131	120	109	106	99	95
157	145	139	131	120	109	106	99
169	157	145	131	131	120	109	106

-128

Blocco scalato

-33	-37	-37	-37	-44	-44	-44	-44
-19	-29	-33	-37	-37	-44	-44	-44
-19	-19	-29	-29	-37	-37	-37	-37
-8	-8	-19	-22	-29	-33	-37	-37
3	3	-8	-19	-22	-29	-33	-37
17	11	3	-8	-19	-22	-29	-33
29	17	11	3	-8	-19	-22	-29
41	29	17	3	3	-8	-19	-22

DCT

Trasformata DCT

-155	101	13	6	2	0	-1	0
-121	-46	-1	-6	1	2	-2	-3
11	0	-1	3	5	3	3	3
-11	1	-3	-3	-2	3	2	0
2	-1	-1	1	-2	-4	0	0
-2	-4	-3	-4	0	4	-1	-4
1	-4	1	-1	-3	-1	1	-4
-1	-3	0	-1	-1	1	-2	0

applico

Matrice di quantizzazione

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

Quantizzazione

Trasformata quantizzata

-10	9	1	0	0	0	0	0
-10	-4	0	0	0	0	0	0
1	0	0	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

Zig-zag

Zig zag	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	-10	9	-10	1	-4	1	0	0	0	-1	EOB

Example

Zig zag	0	1	2	3	4	5	6	7	8	9	10
	-10	9	-10	1	-4	1	0	0	0	-1	EOB

NNNN	Valore	SSSS	Run/Size	Categoria	Valore	Numero di bit
x	-10	4		101	0101	7
0	9	4	0/4	1011	1001	8
0	-10	4	0/4	1011	0101	8
0	1	1	0/1	00	1	3
0	-4	3	0/3	100	011	6
0	1	1	0/1	00	1	3
3	-1	1	3/1	111010	0	7
	EOB			1010		4
					Tot	**46**

Dimensione originale: 8x8x8 bit	512	bit
Dimensione compressa	46	bit
Rapporto di compressione:	8,98	%

Ricostruisco la DCT

-160	99	10	0	0	0	0	0
-120	-48	0	0	0	0	0	0
14	0	0	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

IDCT

Blocco scalato ricostruito

-33	-35	-38	-41	-43	-44	-45	-45
-27	-29	-33	-36	-39	-41	-42	-42
-18	-21	-25	-30	-34	-37	-39	-39
-9	-12	-17	-24	-30	-34	-37	-38
2	-2	-9	-17	-25	-31	-35	-37
14	9	1	-9	-18	-26	-31	-34
27	22	13	2	-9	-18	-24	-27
35	30	21	9	-2	-12	-19	-22

Confronto con l'originale

Errore

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

MSE (Mean Squared Error)	6,86
RMSE (Root Mean Squared Error)	2,62
PSNR (Peak Signal to Noise Ratio)	39,77

Example

DC coefficient differences

Category	Code len.	Code word
0	2	000
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
11	9	111111110

AC coefficients

Run/Size	Code len.	Code word
0/0 (EOB)	4	1010
0/1	2	00
0/2	2	01
0/3	3	100
0/4	4	1011
0/5	5	11010
0/6	7	1111000
0/7	8	11111000
0/8	10	1111110110
0/9	16	1111111110000010
0/A	16	1111111110000011
1/1	4	1100
1/2	5	11011
1/3	7	1111001
1/4	9	111110110
1/5	11	11111110110
1/6	16	1111111110000100
1/7	16	1111111110000101
1/8	16	1111111110000110
1/9	16	1111111110000111
1/A	16	1111111110001000

Example

AC coefficients

Run/Size	Code len.	Code word
2/1	5	11100
2/2	8	11111001
2/3	10	1111110111
2/4	12	111111110100
2/5	16	1111111110001001
2/6	16	1111111110001010
2/7	16	1111111110001011
2/8	16	1111111110001100
2/9	16	1111111110001101
2/A	16	1111111110001110
3/1	6	111010
3/2	9	111110111
3/3	12	111111110101
3/4	16	1111111110001111
3/5	16	1111111110010000
3/6	16	1111111110010001
3/7	16	1111111110010010
3/8	16	1111111110010011
3/9	16	1111111110010100
3/A	16	1111111110010101

AC coefficients

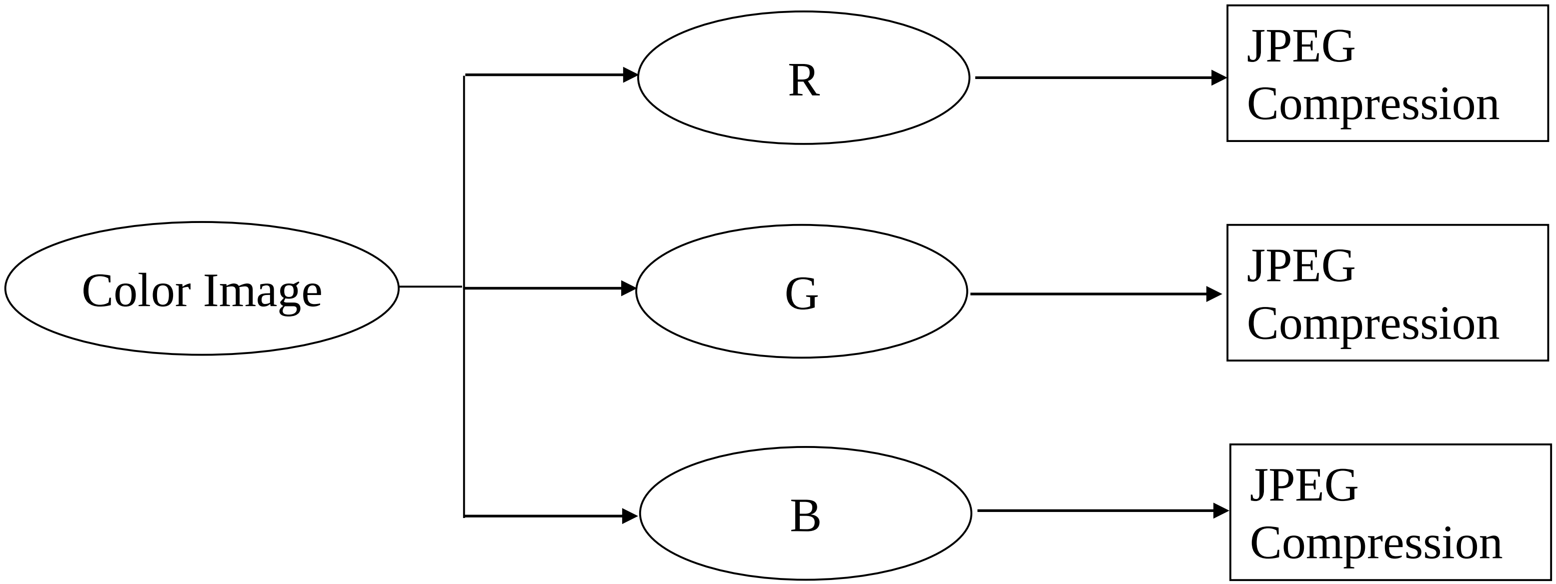
Run/Size	Code len.	Code word
4/1	6	111011
4/2	10	1111111000
4/3	16	1111111110010110
4/4	16	1111111110010111
4/5	16	1111111110011000
4/6	16	1111111110011001
4/7	16	1111111110011010
4/8	16	1111111110011011
4/9	16	1111111110011100
4/A	16	1111111110011101
5/1	7	1111010
5/2	11	11111110111
5/3	16	1111111110011110
5/4	16	1111111110011111
5/5	16	1111111110100000
5/6	16	1111111110100001
5/7	16	1111111110100010
5/8	16	1111111110100011
5/9	16	1111111110100100
5/A	16	1111111110100101

Byte Stuffing

- The syntax of the bit stream is rather complex, but broadly speaking, the concept is that the data is divided into blocks hierarchically ordered. Before each block we have a marker, which allows to have reference points within the file.
- Each marker identifies a segment of file. The marker is followed by two bytes which indicate the length of the segment (measured in bytes). Markers are aligned to the byte and all start with FF (hexadecimal) and then have a specific second byte.
- When, during an encoding, the FF byte is created in the output string, a 00 is also added (the string is "stuffed" with an extra byte, hence the term byte stuffing).
- If after an FF byte the decoder sees a 00, it ignores it. If the byte is not a zero, it must be a marker.
- At the end of the encoding of any block, if you have not obtained a completely full byte, add as many 1s as needed to complete the current byte.

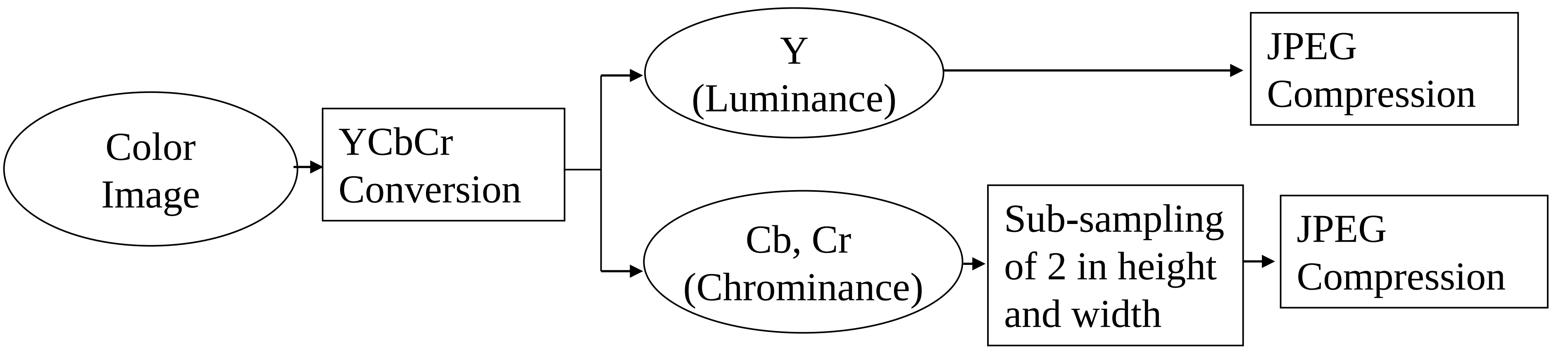
Color Image Management

- If we consider the R, G and B components it **could** be done:



Color Image Management

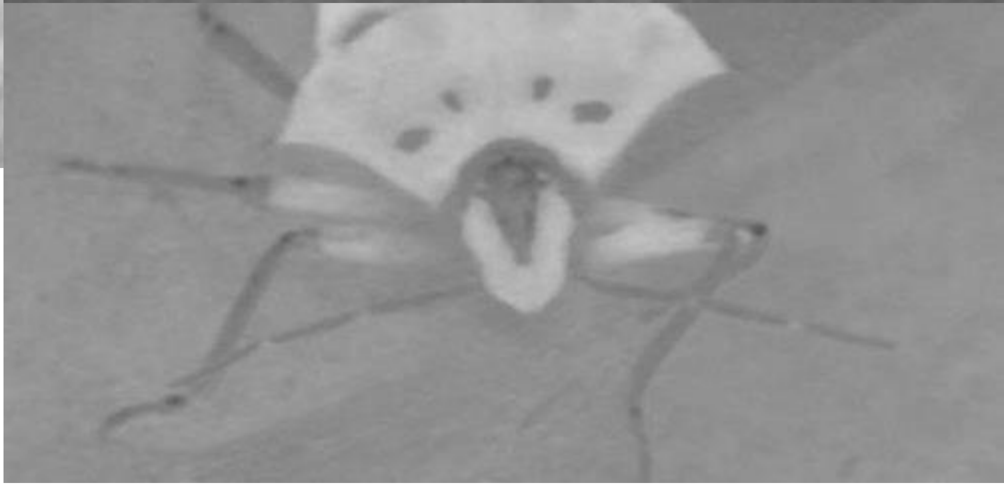
- However, the JFIF format has been defined (JPEG File Interchange Format) which provides the following transformation:



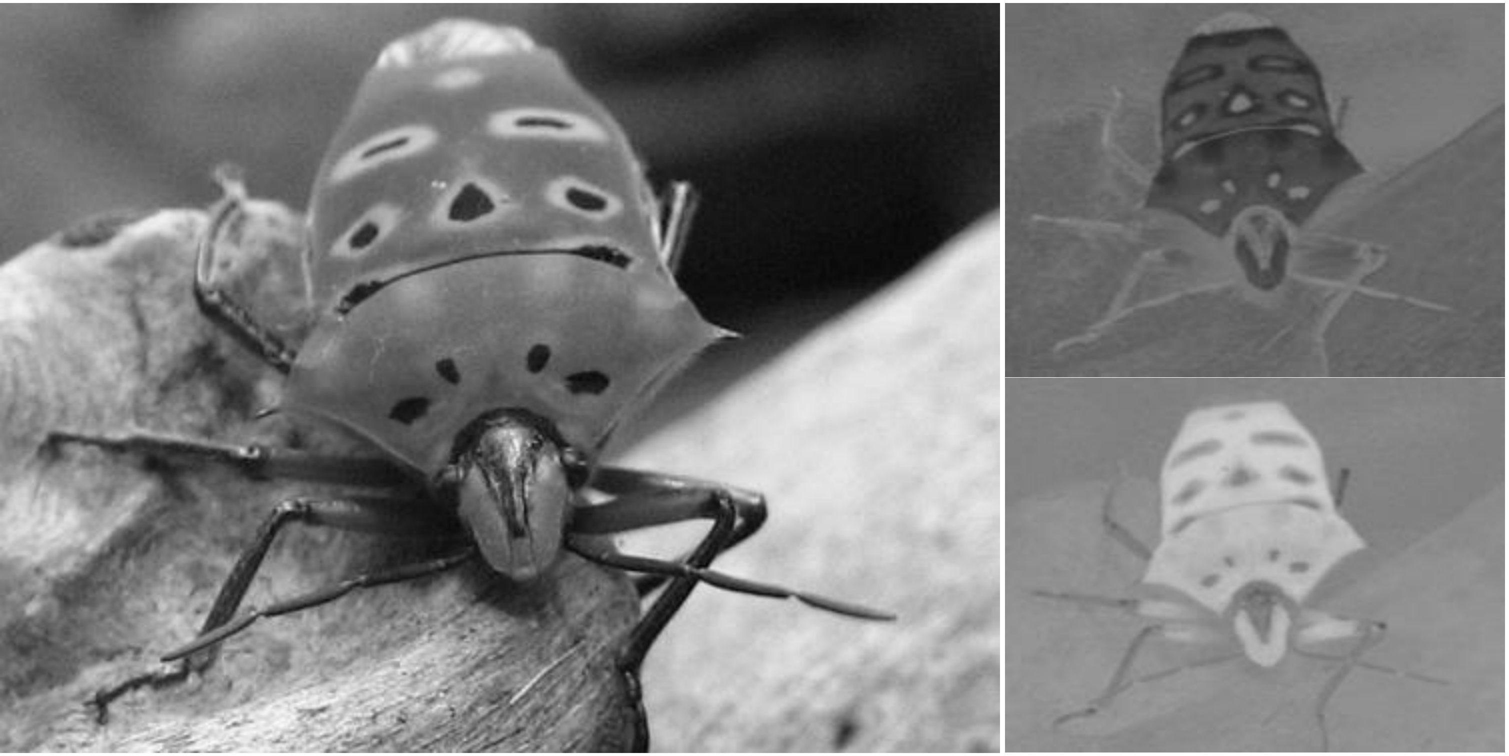
- The color components are sub-sampled by reducing the size to one quarter of the original.

Example of Subsampling in YCbCr

Converting to YCbCr



Normale (1/2) Cb and Cr Subsampling



RGB Reconstruction

Original

Exceeding (1/8) Cb and Cr Subsampling



RGB Reconstruction

Original

Absurd (1/32) Cb and Cr Subsampling

RGB Reconstruction

Original