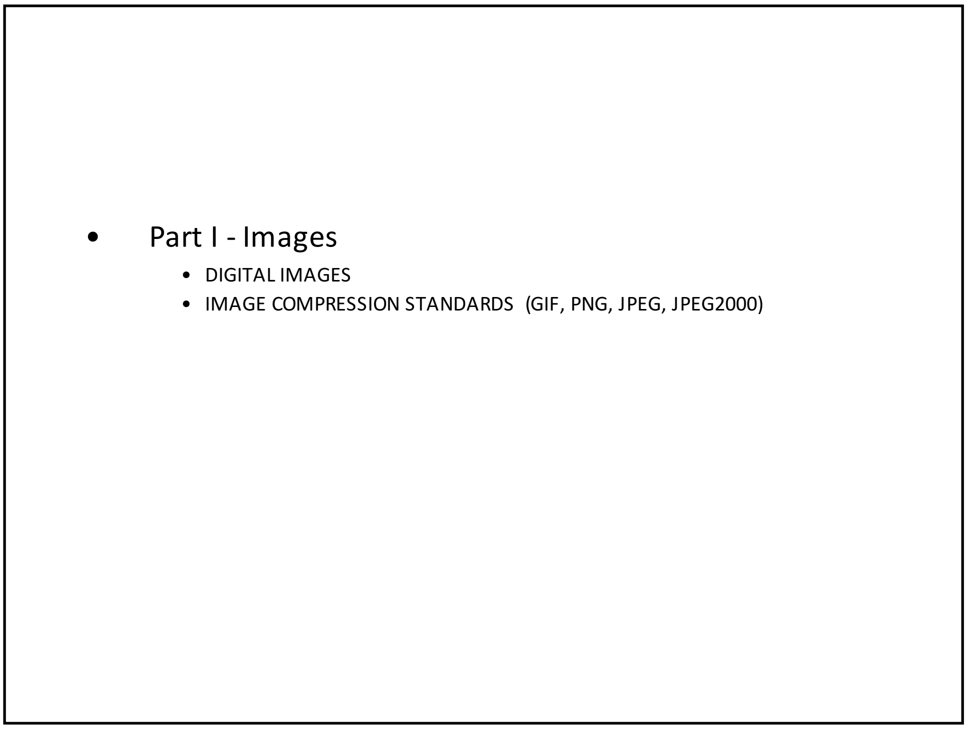


- Part I - Images
  - DIGITAL IMAGES
  - IMAGE COMPRESSION STANDARDS (GIF, PNG, JPEG, JPEG2000)

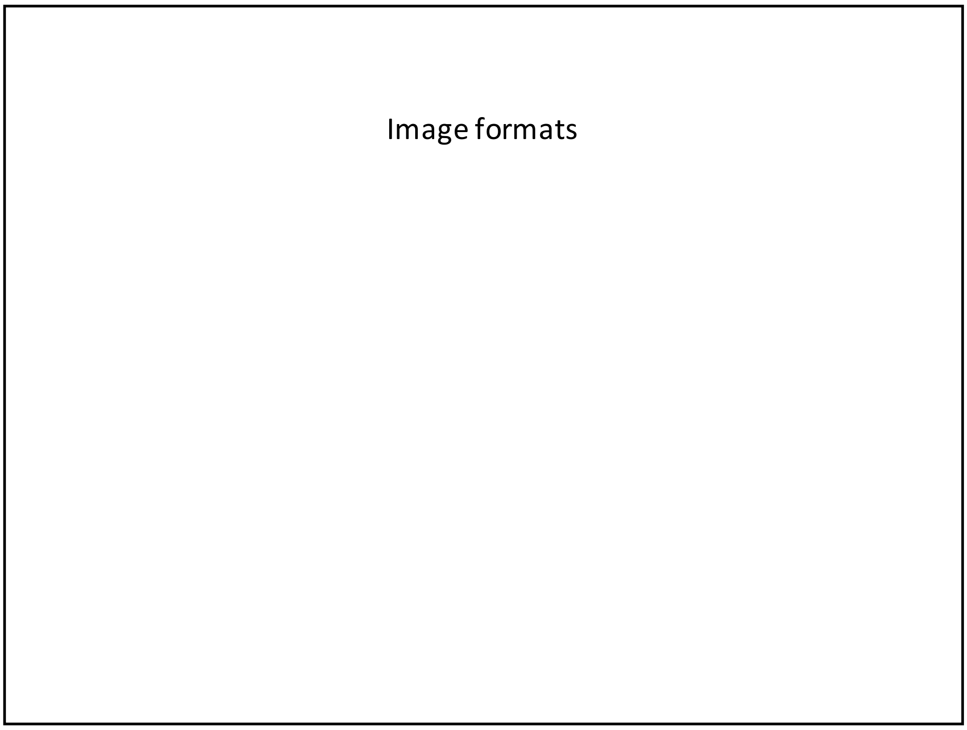


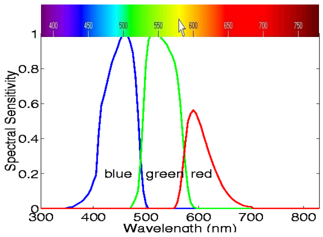
Image formats

Digital images

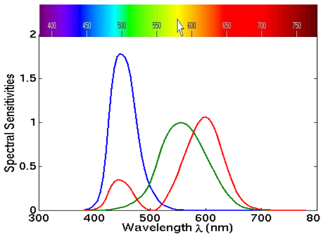
- Digital images are nowadays obtained from photographic digital cameras that use CMOS or CCD sensors (dated) to acquire three color signals in the *red* (R) *green* (G) and *blue* (B) wavelenghts. As a result, the image appears as broken into a series of colored dots called pixels.

- Motivations for RGB sensors originate from mimicking the way in which human eyes operate. The reflected light spectrum can be represented by a 3 element vector, with values which are the proportions of each of the primary colors *red* (R), *green* (G) and *blue* (B) used to produce it. These are the *tristimulus* values.
- Considering Tristimulus: RGB values of camera = Colour * Tristimulus.

Camera Response



Eye Response



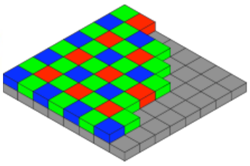
$$R = \int_{\lambda} E(\lambda)\rho_{Skin}(\lambda) f_R(\lambda) d\lambda$$

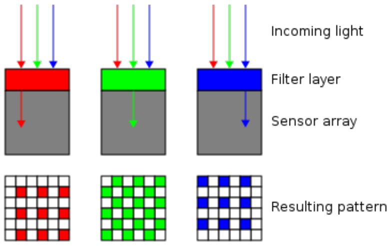
$$G = \int_{\lambda} E(\lambda)\rho_{Skin}(\lambda) f_G(\lambda) d\lambda$$

$$B = \int_{\lambda} E(\lambda)\rho_{Skin}(\lambda) f_B(\lambda) d\lambda$$

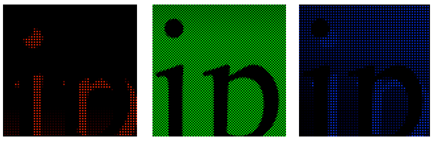
Digital cameras

- Digital cameras often operate with RGB in a *Bayer filter* arrangement: *green* is given twice as many detectors as *red* and *blue* (ratio 1:2:1) in order to achieve higher luminance than chrominance resolution.
- The sensor has a grid of red, green, and blue detectors arranged so that the first row is GBGBGBGB, the next is RGRGRGRG and that sequence is repeated in subsequent rows. For every channel, missing pixels are obtained by interpolation to build up the complete image.





Demosaicing algorithms



Bayer filter samples

original

reconstructed after demosaicing

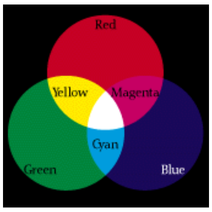
- Due to this, originally, a pixel only records one color out of three and cannot determine the color of the reflected light. An algorithm is needed to estimate for each pixel the color levels for all color components, rather than a single component.

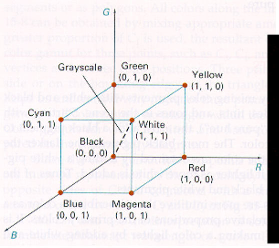
- To obtain a full color image demosaicing algorithms are used that interpolate a set of complete green, red, blue values at each point. This is done in-camera. For example, demosaicing can be done with bilinear interpolation the red value of a non-red pixel is computed as the average of the two or four adjacent red pixels, and similarly for blue and green….

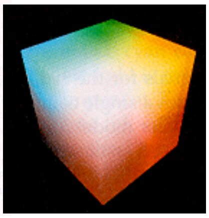
Name	Canon EOS 5D Mark IV	Fujifilm X-T2	Nikon D500	Ricoh GR II	Sony Alpha 6000	Sony Alpha 6300	Sony Cyber-shot DSC-RX100 III	Canon EOS Rebel T6s	Canon PowerShot SX60 HS	Olympus Tough TG-4
Dimensions	4.6 x 5.9 x 3 inches	3.6 x 5.2 x 1.9 inches	4.5 x 5.8 x 3.2 inches	2.5 x 4.6 x 1.4 inches	2.6 x 4.7 x 1.8 inches	2.6 x 4.7 x 1.9 inches	2.3 x 4 x 1.6 inches	4 x 5.2 x 3.1 inches	3.6 x 4.5 x 5 inches	2.6 x 4.4 x 1.2 inches
Weight	1.8 lb	1.1 lb	1.9 lb	7.8 oz	12.1 oz	14.25 oz	10.2 oz	1.2 lb	1.4 lb	8.7 oz
Type	D-SLR	Compact Interchangeable Lens	D-SLR	Compact	Compact Interchangeable Lens	Compact Interchangeable Lens	Compact	D-SLR	Superzoom	Compact
Megapixels	30.4 MP	24 MP	20.9 MP	16 MP	24 MP	24 MP	20 MP	24 MP	16 MP	16 MP
Sensor Size	Full-Frame (24 x 36mm) mm	APS-C (23.6 x 15.6mm) mm	APS-C (15.7 x 23.7mm) mm	APS-C	APS-C (23.5 x 15.6mm) mm	APS-C (23.5 x 15.6mm) mm	1" (13.2 x 8.8mm) mm	APS-C (22.3 x 14.9mm)	1/2.3" (6.2 x 4.6mm) mm	1/2.3" (6.2 x 4.6mm) mm
Maximum ISO	102400	51200	1638400	25600	25600	51200	12800	25600	3200	6400
LCD size	3.2 inches	3 inches	3.2 inches	3 inches	3 inches	3 inches	3 inches	3 inches	3 inches	3 inches
LCD dots	1,620,000	1,040,000	2,359,000	1,229,000	921,600	921,600	1,228,800	1,040,000	922,000	460,000

CIE RGB color space

- A color space is a three-dimensional definition of a color system. The attributes of the color system are mapped onto the coordinate axes of the color space.
- Different color spaces exist: each has advantages and disadvantages for color selection and specification for different applications. The CIE RGB color space is an additive color space, i.e. each color is obtained from the sum of the three primaries, and can be represented as a cube







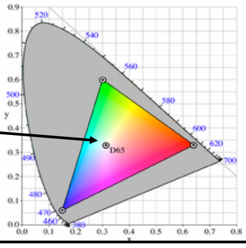
RGB color spaces

- CIE RGB color space is one of the many RGB color spaces, each of which is distinguished by a particular set of monochromatic (single-wavelength) primary colors generated by the hardware. RGB is therefore hardware dependent.
- The sRGB color space (created cooperatively by HP and Microsoft in 1996) is the most widely used in practice. It uses the same primaries as the ITU-R BT.709 primaries, standardized for studio monitors and HDTV and is the reference standard used for monitors, printers and on the Internet. LCDs, digital cameras, printers, and scanners all follow the sRGB standard

Color Space	Gamut	White Point	Primaries					
			xR	yR	xG	yG	xB	yB
– **sRGB , HDTV (ITU-R BT.709),**	**CRT**	**D65**	**0.64**	**0.33**	**0.30**	**0.60**	**0.15**	**0.06**
scRGB	Unlimited	D65	0.64	0.33	0.30	0.60	0.15	0.06
ROMM RGB	Wide	D50	0.7347	0.2653	0.1596	0.8404	0.0366	0.0001
Adobe RGB 98	CRT	D65	0.64	0.33	0.21	0.71	0.15	0.06
Apple RGB	CRT	D65	0.625	0.34	0.28	0.595	0.155	0.07
CIE RGB (1931)	Wide	E	0.7347	0.2653	0.2738	0.7174	0.1666	0.0089
……								

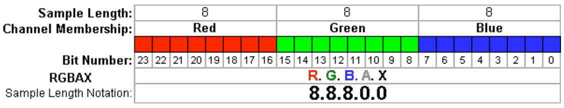
Standard daylight illuminant
White surface illuminated by average midday sun
in western Europe/northern Europe

sRGB gamut



Truecolor representation

- The number of ones and zeros (bits) used to create each pixel denotes the depth of color you can put into your images.
- Truecolor is a method of representing and storing graphical image information. Truecolor defines 256 (2^8) shades of red, green, and blue for each pixel of the digital picture, which results in 256^3 or 16,777,216 (approximately 16.7 million) color variations for each pixel.
- In the absence of any other information, one can generally assume, that any 8-bit-per-channel image file or any 8-bit-per-channel image API or device interface can be treated as being in the sRGB color space.



Sample Length:	8								8								8							
Channel Membership:	Red								Green								Blue							
	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
Bit Number:	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RGBAX	R. G. B. A. X																							
Sample Length Notation:	8.8.8.0.0																							

Image compression

- Compression is the means to have an encoded representation of the image that provides the same (to some extent) appearance with a lower number of bits. Compression can be lossless or lossy. What compression technique is to be used depends on the application. In general:

– Text documents:	lossless compression
– Data for numerical analysis:	lossless compression
– Programs:	lossless compression
– Typographic images:	lossless compression
– WEB images:	lossy compression
– Video:	lossy compression
– Audio:	lossy compression

- Lossy compression guarantees higher compression rates

Lossless compression

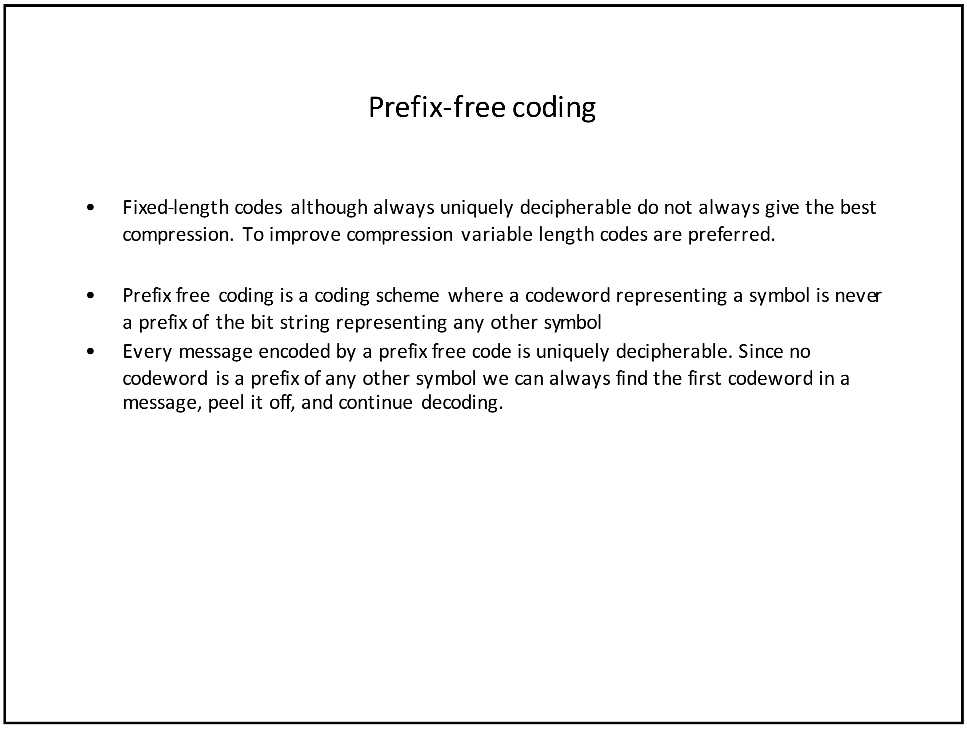
Run-lenght coding

- *Run lenght coding* is a fixed-lenght coding scheme. With run lenght encoding, a sequence of equal symbols is encoded with only one symbol, followed by a number that specifies the times the symbol appears consecutively. A special symbol is required (ex. $).

- For example, if we consider a text string with three character sequences of 11 characters “r”, followed by three sequences of 11 “p” and three sequences of 11 “c” the whole string is encoded as:

  $11r $11r $11r
  $11p $11p $11p
  $11c $11c $11c

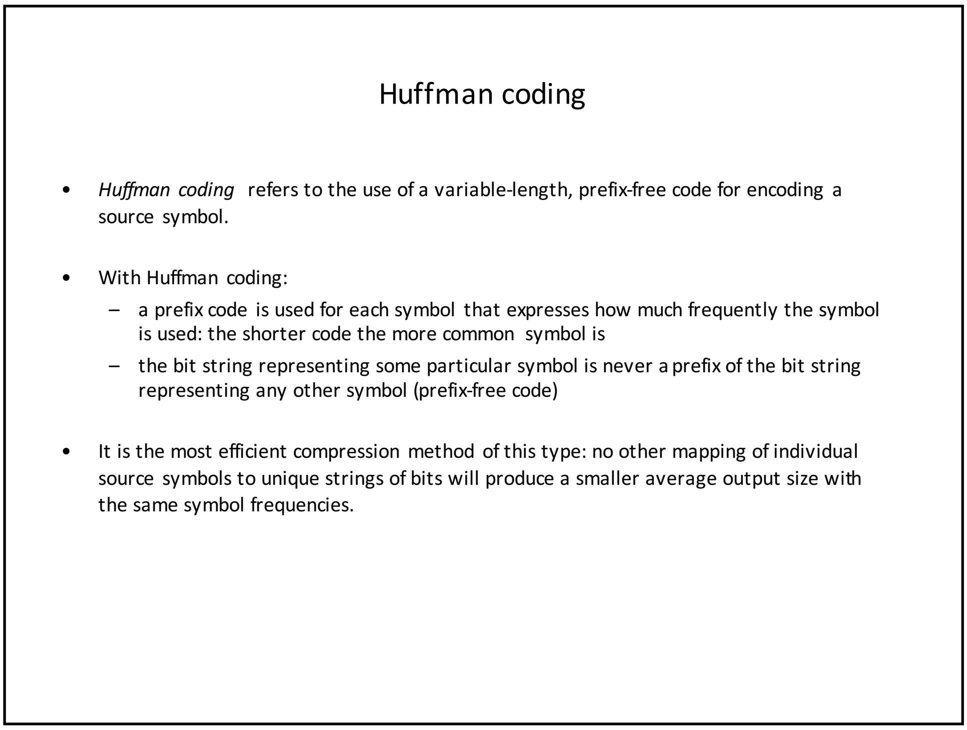
Total characters:	6+6+6 = 18
Total numbers:	3+3+3=9
Total dimension:	27Byte = 216bit



Prefix-free coding

- Fixed-length codes although always uniquely decipherable do not always give the best compression. To improve compression variable length codes are preferred.

- Prefix free coding is a coding scheme where a codeword representing a symbol is never a prefix of the bit string representing any other symbol
- Every message encoded by a prefix free code is uniquely decipherable. Since no codeword is a prefix of any other symbol we can always find the first codeword in a message, peel it off, and continue decoding.



Huffman coding

- *Huffman coding* refers to the use of a variable-length, prefix-free code for encoding a source symbol.

- With Huffman coding:
  - a prefix code is used for each symbol that expresses how much frequently the symbol is used: the shorter code the more common symbol is
  - the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol (prefix-free code)

- It is the most efficient compression method of this type: no other mapping of individual source symbols to unique strings of bits will produce a smaller average output size with the same symbol frequencies.

- The Huffman code for a set of symbols may be generated by constructing a binary tree with nodes containing the symbols to be encoded and their frequencies of occurrence. The tree may be constructed as follows:
  - Step 1. Select the two parentless nodes with the lowest frequencies.
  - Step 2. Create a new node which is the parent of the two lowest frequency nodes.
  - Step 3. Assign the new node a frequency equal to the sum of its children's frequencies.
  - Step 4. Repeat Step 1 until there is only one parentless node left.

- The code for each symbol may be obtained by tracing a path to the symbol from the root of the tree: **1** is assigned for a branch in one direction and **0** is assigned for a branch in the other direction.

- The running time of Huffman's method is fairly efficient, it takes *O* (n log n) operations to construct it.

- Huffman code Example

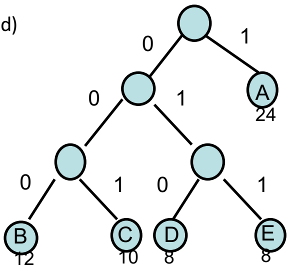
Consider a set of five different symbols. The symbol's frequencies are:

A	24
B	12
C	10
D	8
E	8

If not encoded, this results into a total of 186 bit (3 bit per codeword)

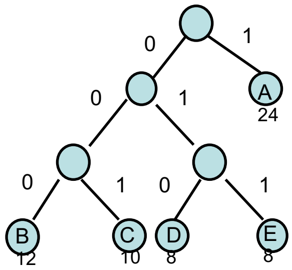
Huffman encoding:

- Step 1. Combine D and E into DE with a frequency of 16
- Step 2. Combine B and C into BC with a frequency of 22
- Step 3. Combine BC and DE into BCDE with a frequency of 38
- Combine A with BCDE into ABCDE with a frequency of 60



- Building the code tree according to Huffman results into a compressed encoding

Symbol	Frequency	Code	Code Length	Total Length
A	24	1	1	24
B	12	000	3	36
C	10	001	3	30
D	8	010	3	24
E	8	011	3	24
186 bit (3 bit code)				138 bit (Huffman encoding)



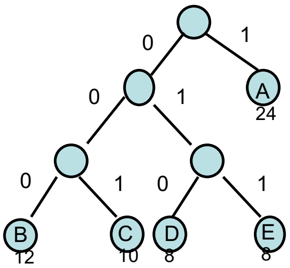
Huffman decoding

- In order to decode Huffman encoded files, the decoding algorithm must know what code was used to encode the data. A table containing symbols and their codes or the Huffman tree can be used.
- Decoding a file is a two step process. First the header data with the table is read in, and the Huffman code for each symbol is reconstructed. Then the encoded data is read and decoded.

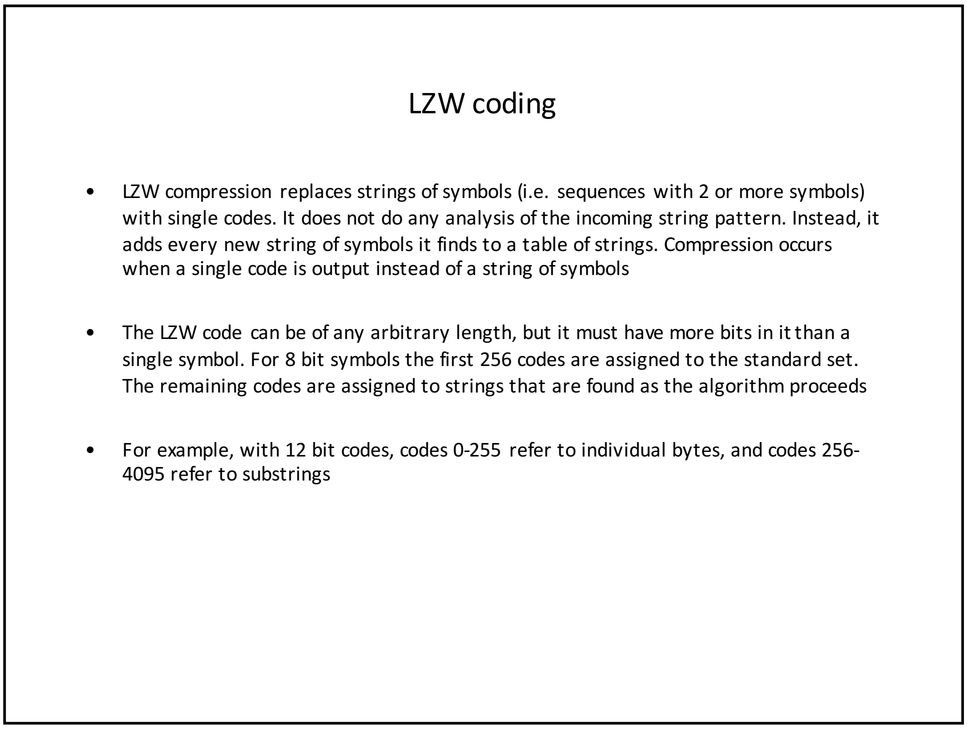
- The fastest method for decoding symbols is to read the encoded file one bit at time and traverse the Huffman tree according to each of the bits until a leaf containing a symbol is reached. When a bit causes a leaf of the tree to be reached, the symbol contained in that leaf is written to the decoded file, and traversal starts again from the root of the tree.

- Example

Input sequence	1**011**000**010**011
Decoded sequence	A E B D E



- Although Huffman's original algorithm is optimal for a stream of unrelated symbols with a known input probability distribution, it is not optimal when the symbol-by-symbol restriction is dropped, or when the probability mass functions are unknown, …..
- Other methods such as f.e. *LZW coding* used in GIF images often have better compression capability
- These methods can combine an arbitrary number of symbols for more efficient coding, and generally adapt to the actual input statistics when input probabilities are not precisely known or vary significantly within the stream.

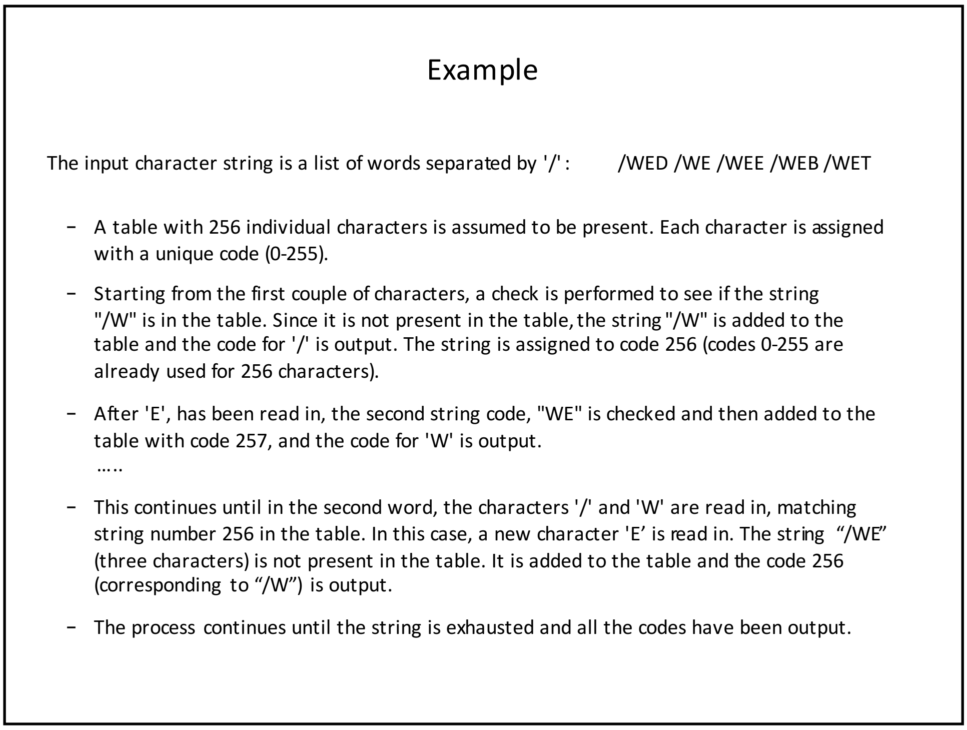


LZW coding

- LZW compression replaces strings of symbols (i.e. sequences with 2 or more symbols) with single codes. It does not do any analysis of the incoming string pattern. Instead, it adds every new string of symbols it finds to a table of strings. Compression occurs when a single code is output instead of a string of symbols

- The LZW code can be of any arbitrary length, but it must have more bits in it than a single symbol. For 8 bit symbols the first 256 codes are assigned to the standard set. The remaining codes are assigned to strings that are found as the algorithm proceeds

- For example, with 12 bit codes, codes 0-255 refer to individual bytes, and codes 256-4095 refer to substrings

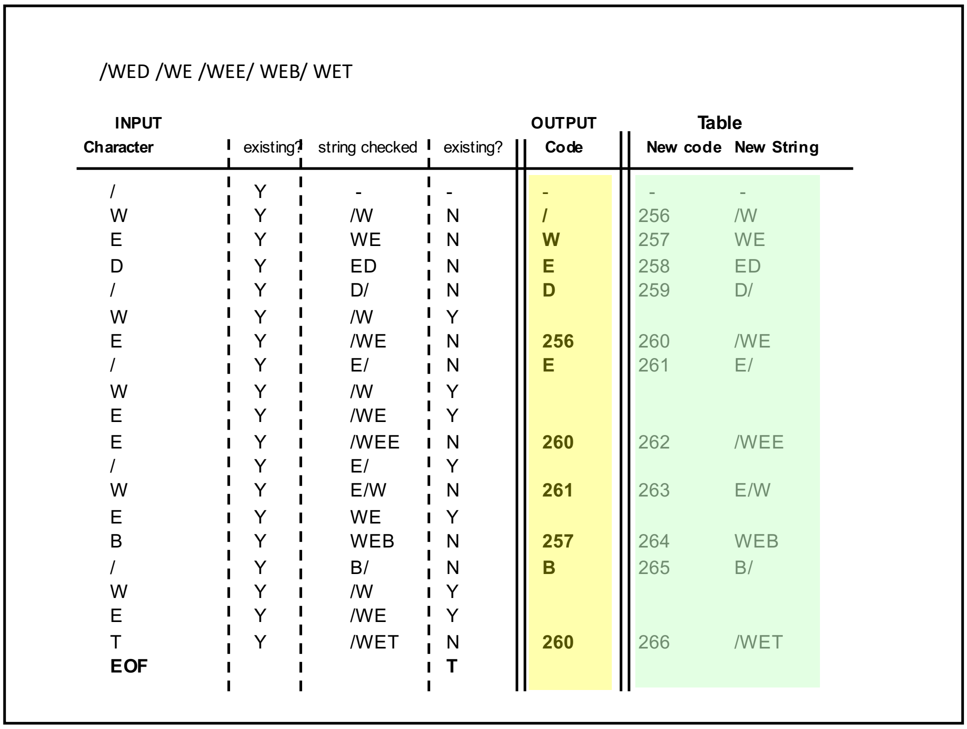


Example

The input character string is a list of words separated by '/' : /WED /WE /WEE /WEB /WET

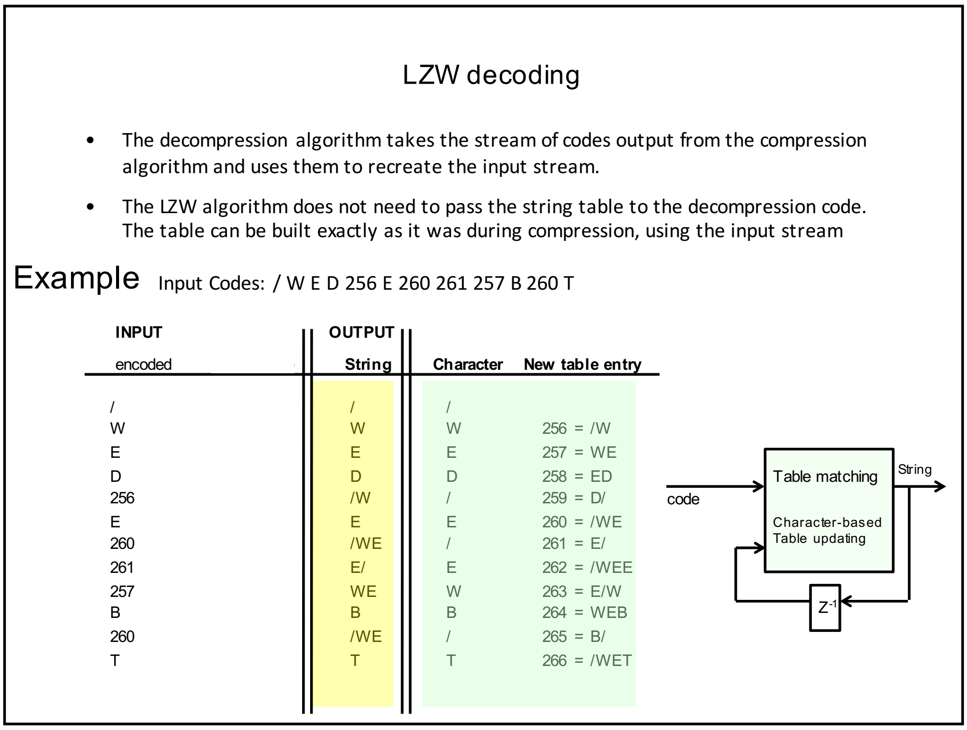
- A table with 256 individual characters is assumed to be present. Each character is assigned with a unique code (0-255).
- Starting from the first couple of characters, a check is performed to see if the string "/W" is in the table. Since it is not present in the table, the string "/W" is added to the table and the code for '/' is output. The string is assigned to code 256 (codes 0-255 are already used for 256 characters).
- After 'E', has been read in, the second string code, "WE" is checked and then added to the table with code 257, and the code for 'W' is output.

  …..
- This continues until in the second word, the characters '/' and 'W' are read in, matching string number 256 in the table. In this case, a new character 'E' is read in. The string "/WE" (three characters) is not present in the table. It is added to the table and the code 256 (corresponding to "/W") is output.
- The process continues until the string is exhausted and all the codes have been output.



/WED /WE /WEE/ WEB/ WET

INPUT Character	existing?	string checked	existing?	OUTPUT Code	Table New code	 New String
/	Y	-	-	-	-	-
W	Y	/W	N	/	256	/W
E	Y	WE	N	**W**	257	WE
D	Y	ED	N	**E**	258	ED
/	Y	D/	N	**D**	259	D/
W	Y	/W	Y			
E	Y	/WE	N	**256**	260	/WE
/	Y	E/	N	**E**	261	E/
W	Y	/W	Y			
E	Y	/WE	Y			
E	Y	/WEE	N	**260**	262	/WEE
/	Y	E/	Y			
W	Y	E/W	N	**261**	263	E/W
E	Y	WE	Y			
B	Y	WEB	N	**257**	264	WEB
/	Y	B/	N	**B**	265	B/
W	Y	/W	Y			
E	Y	/WE	Y			
T	Y	/WET	N	**260**	266	/WET
EOF			**T**			



LZW decoding

- The decompression algorithm takes the stream of codes output from the compression algorithm and uses them to recreate the input stream.
- The LZW algorithm does not need to pass the string table to the decompression code. The table can be built exactly as it was during compression, using the input stream

Example Input Codes: / W E D 256 E 260 261 257 B 260 T

INPUT encoded	OUTPUT String	 Character	 New table entry
/	/	/	
W	W	W	256 = /W
E	E	E	257 = WE
D	D	D	258 = ED
256	/W	/	259 = D/
E	E	E	260 = /WE
260	/WE	/	261 = E/
261	E/	E	262 = /WEE
257	WE	W	263 = E/W
B	B	B	264 = WEB
260	/WE	/	265 = B/
T	T	T	266 = /WET

GIF image representation standard

- *GIF Graphic Interchange Format* is an 8 bit per pixel image format using a palette of up to 256 distinct colors from the 24-bit RGB color space. A GIF image employs *lossless LZW compression* so that the file size of an image may be reduced without degrading the visual quality, provided the image can be rendered with only 256 colours. Its lossless compression preserves very sharp edges.

- The 256 colors limitation makes the GIF format unsuitable for color photographs and other images with continuous color. It is instead well-suited for simpler images such as graphics, sharp-edged line art or logos with solid areas of color with a very limited number of colors.
- Monochrome photographs (with continuous grey tones) can be represented well as GIFs but have large file sizes due to the inappropriate compression technique

- Formerly widely used on the web, PNG has increasingly replaced GIFs on web pages.

GIF animation

- GIF89a animation provides the ability to set the cell's (technically called an "*animation frame*") movement speed in 1/100 of a second. An internal clock embedded into the GIF keeps count and flips the image when the time comes.

- GIF89a was designed based on the principle of rendering images to a logical screen. Each image could optionally have its own palette, and the format allows to specify delay and waiting for user input

GIF Transparency

- In transparent GIF the computer is told to hone in on one color. A particular red / green / blue shade already found in the image is chosen and blanked out (the color is dropped from the palette that makes up the image), so that whatever is behind it shows through. A transparent GIF is limited in that only one color of the 256-shade palette can be made transparent
- The process is similar to *chroma key* used in television. A computer is told to hone in on a specific color, let's say it's green. Chroma key screens are usually green because it's the color least likely to be found in human skin tones. That chroma is then erased and replaced by another image.

PNG image representation standard

- *PNG Portable Network Graphics* is a lossless compression format created to improve and replace the GIF format. It overcomes the GIF limitation to 256 colors.
- PNG was designed for transfering images on the internet, not professional graphics, and so does not support other color spaces than RGB. It is the optimal choice for exporting images with repeating gradients for web usage.
- PNG is also useful for saving temporary photographs that require successive editing. When the photograph is ready to be distributed, it can then be saved as a JPEG (leter), and this limits the information loss to just one generation. However PNG does not support EXIF (Exchangeable Image File) image data (including camera settings like shutter speed, focal lenght, exposure compensation, flash used… and scene information, date and time…) from sources such as digital cameras, which makes it problematic for use amongst amateur and especially professional photographers.
- PNG is a single-image format. A companion format called MNG has been defined for animation. PNG files use file-extension "PNG" or "png”

Deflate coding

- PNG employs the DEFLATE lossless compression algorithm (also used in the PKZIP archiving tool and specified in RFC 1951). A DEFLATE stream consists of a series of blocks. Each block uses a single mode of compression and is preceded by a header.

- Compression is achieved through two steps:
  - *LZ77 compression algorithm*: matching and replacement of duplicate strings with pointers: if a repeated string of bits exists a back reference to the previous location of the same string is inserted.
  - *Huffman coding*: replacing symbols with new, weighted symbols based on frequency of use

The LZ77 compression

- The LZ77 compression finds sequences of characters that are repeated. It uses a sliding window of 32K (records what the last 32768 characters were).

- When the same sequence of characters is encountered the sequence is replaced by:
  - ***distance*** (how far back in the window)
  - ***lenght*** (the number of identical characters),

  which is equivalent to the statement: "*each of the* ***next length*** *characters is equal to the characters exactly* ***distance characters behind*** *it in the uncompressed stream*"

Example

- Let’s take the sequence : *Home home home home home*
  - Consider the characters *Home h* and the next 5 characters ***ome h*** (in bold):
    *Home h**ome h**ome home home*
  - There is an exact match of the last 5 characters ***ome h*** with the characters before, 5 positions behind the current point. We can output special characters to the stream that represent a *number for length*, and a *number for distance*. Exactly we can encode:
    ***Home h* [D=5, L=5]**
  - Considering also the characters that follow each of the two strings declared to be equal we see that other characters are the same. We can further increase compression as:
    ***Home h* [D=5, L=18]**

- PNG supports indexed palette-based (palettes of 24-bit RGB colors) or grayscale or RGB images (one or more channels). The number of channels will depend on whether the image is greyscale or color and whether it has an alpha channel.

- PNG allows the following combinations of channels:
  - indexed (channel containing indexes into a palette or colors)
  - greyscale
  - greyscale and alpha (0/1 indicates the level of transparency for each pixel)
  - red, green and blue (rgb / truecolor)
  - red, green, blue and alpha

Type	**Bit depth per channel**				
	1	**2**	**4**	**8**	**16**
indexed (color type 3)	1	2	4	8	No
greyscale (color type 0)	1	2	4	8	16
greyscale & alpha (color type 4)	No	No	No	16	32
Truecolor (RGB - color type 2)	No	No	No	24	48
Truecolor & alpha (RGBA - color type 6)	No	No	No	32	64

cell values are total bits per pixel

Lossy compression

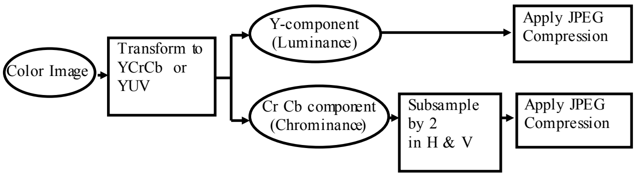
The JPEG Standard

- *Lossy JPEG* is a uses *transform coding*. It employs a compression algorithm developed by the Joint Photographic Experts Group JPEG that trades-off detail in the displayed picture for a smaller storage file (http://www.jpeg.org/).
- JPEG is the format most used for storing and transmitting photographs on the web. It is preferred to formats such as GIF, which has a limit of 256 distinct colors that is insufficient for colour photographs, and PNG, which produces much larger image files for this type of image.

- The compression algorithm is *not* well suited for line drawings and other textual or iconic graphics. The PNG and GIF formats are preferred for these types of images.
- JPEG specifies both the codec defining how an image is transformed into a stream of bytes, and the file format JFIF (JPEG File Intechange Format) used to contain that stream.

JPEG for color images

- With JPEG the image should be converted from RGB into a different color space called YCbCr or Y'CbCr.
- It allows greater perceptual image quality for the same compression because the brightness information, which is more important to the eventual perceptual quality of the image, is confined to a single channel. JPEG is applied to each channel separately (downsampling the chroma channel)


Color Image

Transform to YCrCb or YUV

Y-component (Luminance)

Apply JPEG Compression

Cr Cb component (Chrominance)

Subsample by 2 in H & V

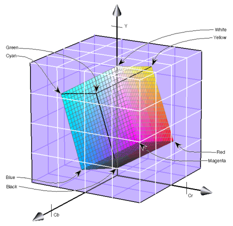
Apply JPEG Compression

- .

YCbCr color space

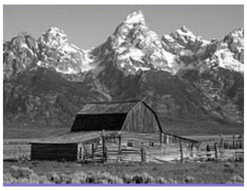
- YCbCr or Y'CbCr is a family of color spaces used in video and digital photography systems:
  - Y (Y') is the *luminance* (*luma*) component and is obtained as a combination of R G and B. The Y' *luma* distinguishes from Y *luminance*, in that light intensity is non-linearly encoded using gamma.
  - Cb and Cr are the *blue-difference* and *red-difference* chroma components and are obtained instead subtracting Y from R and B signals respectively
- Colors are distorted passing from RGB to YCbCr or Y'CbCr. The results are scaled and rounded, and offsets are typically added. A particular conversion to $Y'C_BC_R$ is specified in the JFIF standard, and should be performed for the resulting JPEG file to have maximum compatibility

$$\begin{aligned}
Y' &= \quad 0 + (0.299 \quad \cdot R'_D) + (0.587 \quad \cdot G'_D) + (0.114 \quad \cdot B'_D)\\
C_B &= 128 - (0.168736 \cdot R'_D) - (0.331264 \cdot G'_D) + (0.5 \quad \cdot B'_D)\\
C_R &= 128 + (0.5 \quad \cdot R'_D) - (0.418688 \cdot G'_D) - (0.081312 \cdot B'_D)
\end{aligned}$$

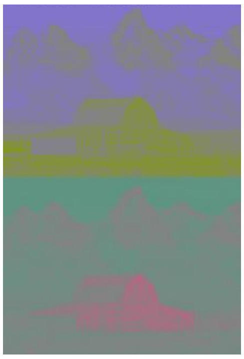


Y’CbCr color space

Original image



Y’



Cb

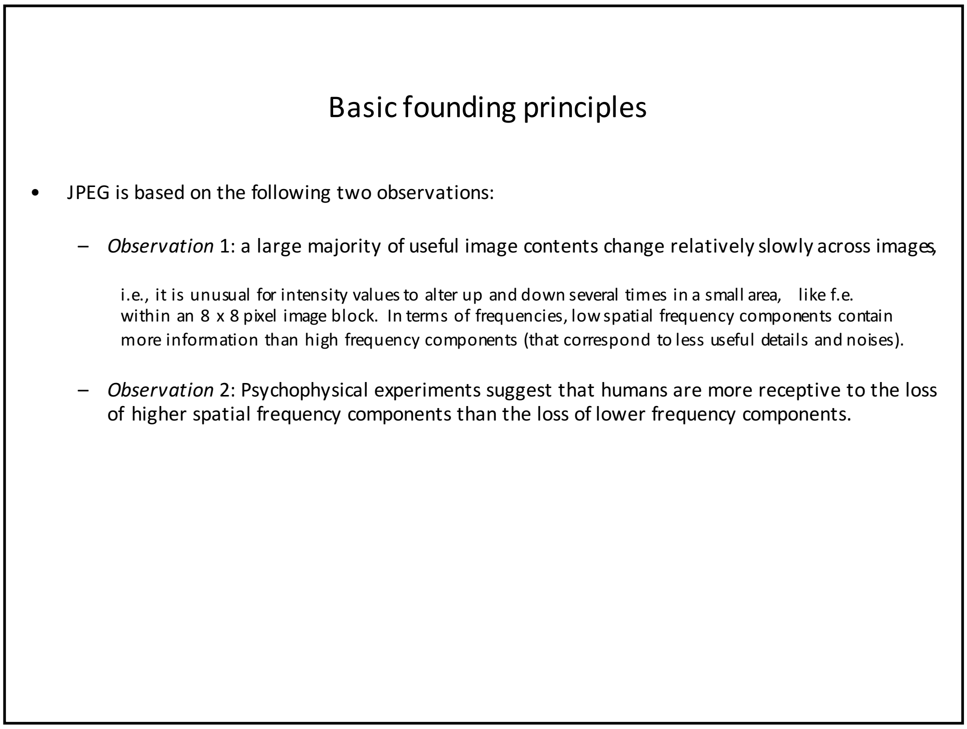
Cr

Some rules of use

- A good rule is to save your JPEGs at 50% or medium compression. 50% compression means that 50% of the image is included in the algorithm. The difference between the 1% and 50% compression is not too bad, but the drop in bytes is impressive.

.

- Since JPEG is lossy, bytes are lost at the expense of detail. It is visible where the compression algorithm found groups of pixels that are close in color and grouped them all together as one.

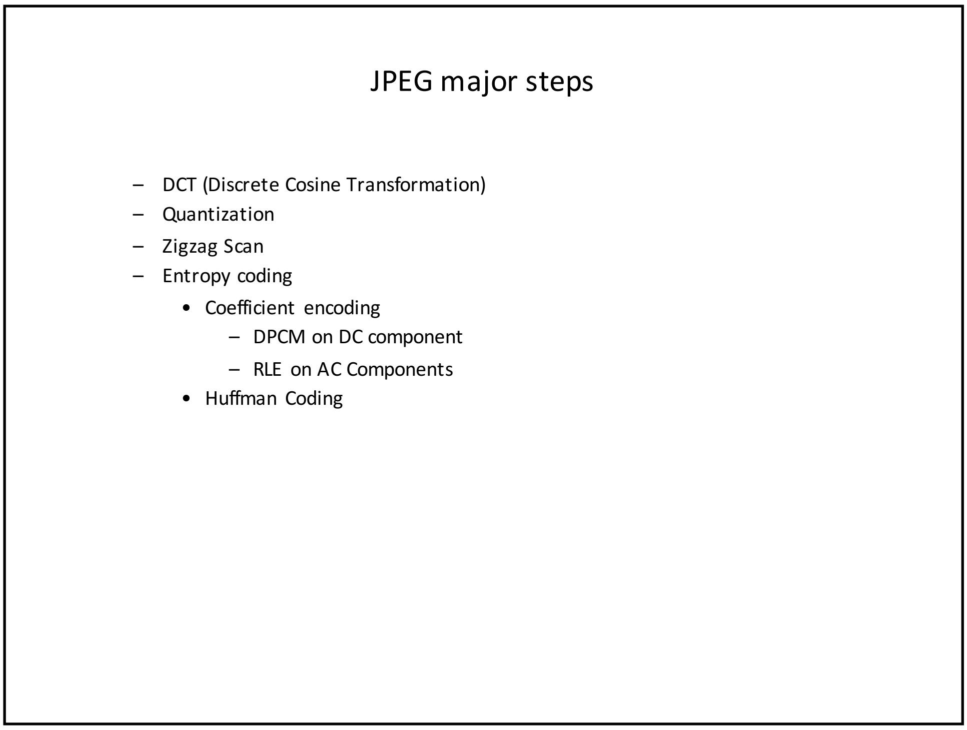


Basic founding principles

- JPEG is based on the following two observations:
  - *Observation* 1: a large majority of useful image contents change relatively slowly across images,

    i.e., it is unusual for intensity values to alter up and down several times in a small area, like f.e. within an 8 x 8 pixel image block. In terms of frequencies, low spatial frequency components contain more information than high frequency components (that correspond to less useful details and noises).

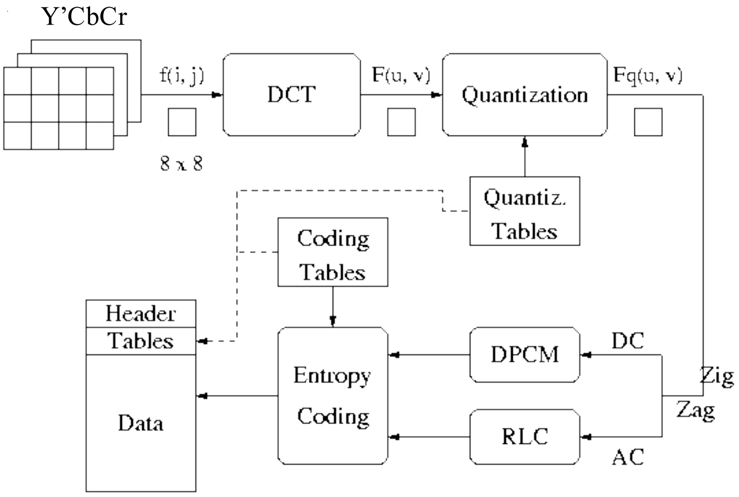
  - *Observation* 2: Psychophysical experiments suggest that humans are more receptive to the loss of higher spatial frequency components than the loss of lower frequency components.



JPEG major steps

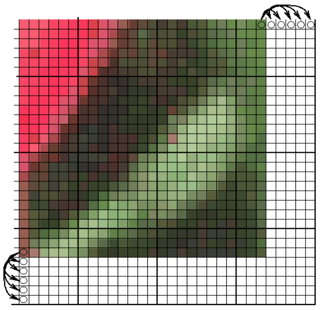
- DCT (Discrete Cosine Transformation)
- Quantization
- Zigzag Scan
- Entropy coding
  - Coefficient encoding
    - DPCM on DC component
    - RLE on AC Components
  - Huffman Coding

JPEG chain

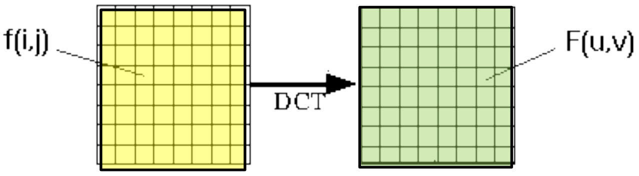


DCT (Discrete Cosine Transformation)

- Apply DCT to 8x8 image blocks
- If the image size is not a multiple of 8, then add copies of the last row or column until a multiple of 8 is reached. This makes both tone and luminance of the 8x8 block not change too much after DCT, as it would be if these elements were set to 0.



- DCT allows to shift from spatial domain to frequency domain:



  - f(i,j) is the value that is present in the (i,j) position of the 8x8 block of the original image
  - F(u,v) is the DCT coefficient in the (u,v) position of the 8x8 matrix that encodes the transformed coefficients.

Discrete Cosine Transform (DCT)

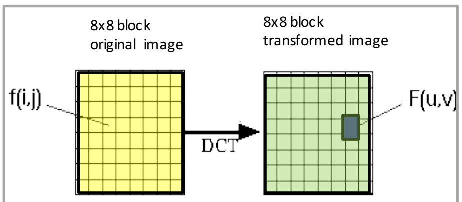
$$F(u,v) = \frac{\Lambda(u)\Lambda(v)}{4}\sum_{i=0}^{7}\sum_{j=0}^{7}\cos\frac{(2i+1)\cdot u\pi}{16}\cdot\cos\frac{(2j+1)\cdot v\pi}{16}\cdot f(i,j)$$

$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & for\ \xi = 0 \\ 1 & otherwise \end{cases}$$

Inverse Discrete Cosine Transform (IDCT)

$$\tilde{f}(i,j) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7}\Lambda(u)\Lambda(v)\cos\frac{(2i+1)\cdot u\pi}{16}\cdot\cos\frac{(2j+1)\cdot v\pi}{16}\cdot F(u,v)$$

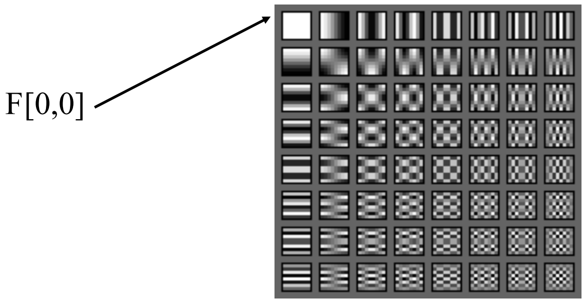
$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & for\ \xi = 0 \\ 1 & otherwise \end{cases}$$



F(u,v) is the DCT coefficient in the (u,v) position of the 8x8 matrix that encodes the transformed coefficients

f(i,j) is the value that is present in the (i,j) position of the 8x8 block of the original image

The 64 (8 x 8) DCT basis functions



Quantization

- To reduce number of bits per sample, quantization is used:

  F'[u, v] = round (F[u, v] / q[u, v])

  where q(u,v) is the quantization matrix and F(u,v) is the DCT coefficient matrix.

  Example: 101101 = 45 (6 bits)
  q[u, v] = 4 ⇨ 11= 1011

- Quantization error is the main source of the lossy compression. Different quantization matrices can be used.

- *Uniform Quantization*
  Each F[u,v] is divided by the same constant *N*.

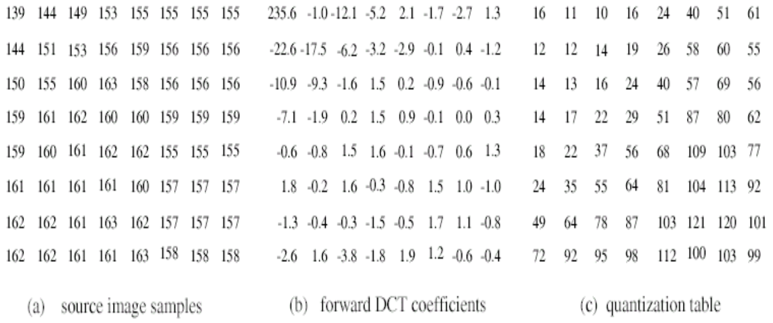
- *Non-uniform Quantization*
  accounts for the fact that human eye is most sensitive to low frequencies (upper left corner), less sensitive to high frequencies (lower right corner), more sensitive to luminance, less to color

Luminance Quantization Table q(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

Chrominance Quantization Table q(u, v)

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



139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

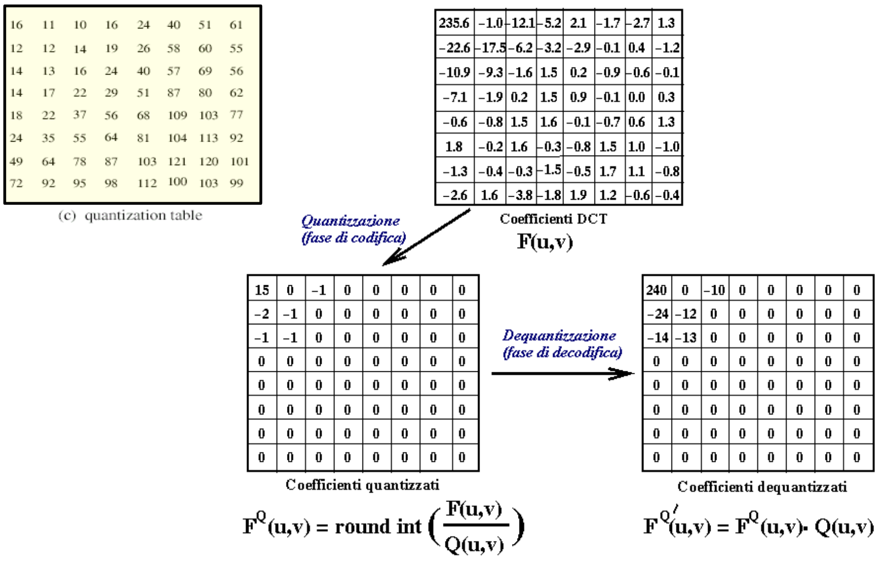
(a) source image samples

235.6	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	0.9	-0.1	0.0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

(b) forward DCT coefficients

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

(c) 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

(c) quantization table

235.6	−1.0	−12.1	−5.2	2.1	−1.7	−2.7	1.3
−22.6	−17.5	−6.2	−3.2	−2.9	−0.1	0.4	−1.2
−10.9	−9.3	−1.6	1.5	0.2	−0.9	−0.6	−0.1
−7.1	−1.9	0.2	1.5	0.9	−0.1	0.0	0.3
−0.6	−0.8	1.5	1.6	−0.1	−0.7	0.6	1.3
1.8	−0.2	1.6	−0.3	−0.8	1.5	1.0	−1.0
−1.3	−0.4	−0.3	−1.5	−0.5	1.7	1.1	−0.8
−2.6	1.6	−3.8	−1.8	1.9	1.2	−0.6	−0.4

Coefficienti DCT

$F(u,v)$

Quantizzazione (fase di codifica)

15	0	−1	0	0	0	0	0
−2	−1	0	0	0	0	0	0
−1	−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	0	0	0	0	0	0	0

Coefficienti quantizzati

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

Dequantizzazione (fase di decodifica)

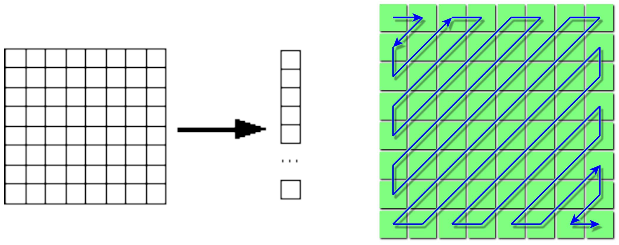
240	0	−10	0	0	0	0	0
−24	−12	0	0	0	0	0	0
−14	−13	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Coefficienti dequantizzati

$$F^{Q'}(u,v) = F^Q(u,v) \cdot Q(u,v)$$

Zig-zag scan

- As a result of quantization we have a 8x8 matrix with many elements equal to 0. Non null coefficients are all in the upper left corner.
- This suggests to transform the 8x8 matrix into a 64 element vector using a zig-zag order. Zig-Zag scan is used to group low frequency coefficients in the top of the vector: *maps 8x8 to 1x64 vector*



Coefficient encoding

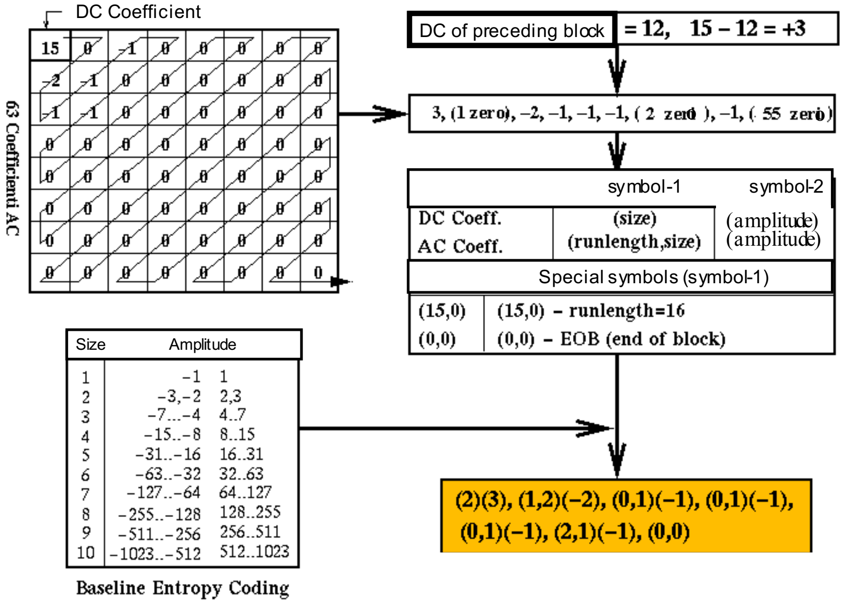
- Coefficients are encoded differently:
  - Differential Pulse Code Modulation (DPCM) on the DC component
    - DC component is large and varied, but often close to the value of the DC component of the previous block.
    - According to this JPEG encodes the difference between the previous and the current 8 x 8 block (DC diff)
  - Run Length Encoding (RLE) on AC components
    - Many of the AC coefficients are equal to 0.
    - According to this they are encoded using RLE, which counts the number of consecutive 0s. A minimum of 0 to a maximum of 16 consecutive 0s is allowed. In the latter case the special symbol (15,0) is used. The end of block is encoded with (0,0).

- For the DC component (DC diff) we build the pair: (**SIZE**) (**AMPLITUDE**)
  SIZE is the number of bits needed to represent the DC difference value; AMPLITUDE is the value of the DC difference .

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

Example: if DC value is 4, 3 bits are needed.

- For the AC components the following representation (**RUNLENGHT**, **SIZE**) (**AMPLITUDE**) is used where RUNLENGHT is the number of consecutive 0 (from 0 to 15), SIZE has the same meaning as for the DC coefficient, AMPLITUDE is the actual value for nonzero AC coefficients



15	0	−1	0	0	0	0	0
−2	−1	0	0	0	0	0	0
−1	−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	0	0	0	0	0	0	0

DC of preceding block = 12, 15 − 12 = +3

3, (1 zero), −2, −1, −1, −1, (2 zero), −1, (55 zero)

	symbol-1	symbol-2
DC Coeff.	(size)	(amplitude)
AC Coeff.	(runlength,size)	(amplitude)

Special symbols (symbol-1)

(15,0)	(15,0) – runlength=16
(0,0)	(0,0) – EOB (end of block)

Size	Amplitude	
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..−64	64..127
8	−255..−128	128..255
9	−511..−256	256..511
10	−1023..−512	512..1023

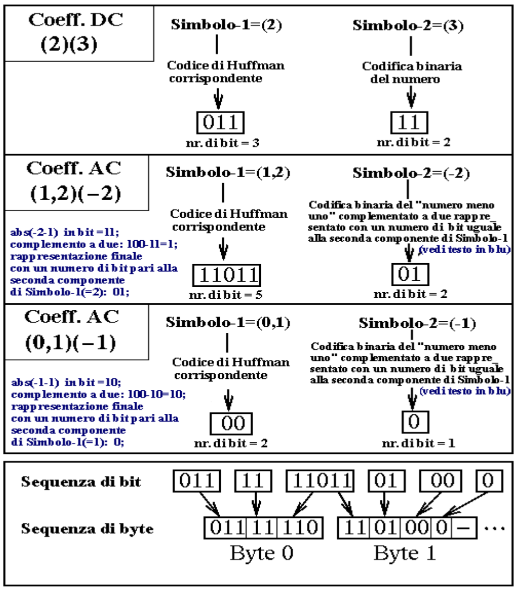
Baseline Entropy Coding

(2)(3), (1,2)(−2), (0,1)(−1), (0,1)(−1), (0,1)(−1), (2,1)(−1), (0,0)

Huffman encoding

- After we have encoded every block, we have a sequence of symbols:
  - Symbol 1: (SIZE) or (RLE, SIZE)
  - Symbol 2: (AMPLITUDE)
- These symbols are further encoded using the Huffman encoding to reduce the number of data.
- Most frequent symbols are encoded with shorter codes. Less frequent with longer ones. Huffman tables provide codes for every symbol of the sequence. Huffman tables are different for DC and AC symbol 1.
- Huffman Tables can be custom (sent in header) or default.

Symbol1 and Symbol2 encoding



Coeff. DC (2)(3)

Simbolo-1=(2)

Codice di Huffman corrispondente

011

nr. di bit = 3

Simbolo-2=(3)

Codifica binaria del numero

11

nr. di bit = 2

Coeff. AC (1,2)(−2)

abs(-2-1) in bit =11; complemento a due: 100-11=1; rappresentazione finale con un numero di bit pari alla seconda componente di Simbolo-1(=2): 01;

Simbolo-1=(1,2)

Codice di Huffman corrispondente

11011

nr. di bit = 5

Simbolo-2=(-2)

Codifica binaria del "numero meno uno" complementato a due rappre_sentato con un numero di bit uguale alla seconda componente di Simbolo-1 (vedi testo in blu)

01

nr. di bit = 2

Coeff. AC (0,1)(−1)

abs(-1-1) in bit =10; complemento a due: 100-10=10; rappresentazione finale con un numero di bit pari alla seconda componente di Simbolo-1(=1): 0;

Simbolo-1=(0,1)

Codice di Huffman corrispondente

00

nr. di bit = 2

Simbolo-2=(-1)

Codifica binaria del "numero meno uno" complementato a due rappre_sentato con un numero di bit uguale alla seconda componente di Simbolo-1 (vedi testo in blu)

0

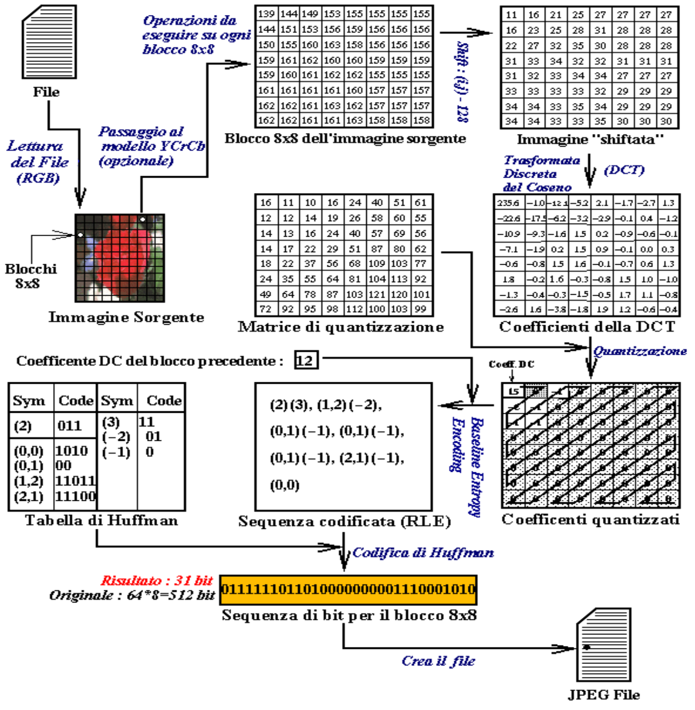
nr. di bit = 1

Sequenza di bit: 011 11 11011 01 00 0

Sequenza di byte: 011 11 110 | 11 01 00 0 – ...

Byte 0 Byte 1

Byte stuffing



Blocco 8x8 dell'immagine sorgente

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

Immagine "shiftata"

11	16	21	25	27	27	27	27
16	23	25	28	31	28	28	28
22	27	32	35	30	28	28	28
31	33	34	32	32	31	31	31
31	32	33	34	34	27	27	27
33	33	33	33	32	29	29	29
34	34	33	35	34	29	29	29
34	34	33	33	35	30	30	30

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

Coefficienti della DCT

235.6	−1.0	−12.1	−5.2	2.1	−1.7	−2.7	1.3
−22.6	−17.5	−6.2	−3.2	−2.9	−0.1	0.4	−1.2
−10.9	−9.3	−1.6	1.5	0.2	−0.9	−0.6	−0.1
−7.1	−1.9	0.2	1.5	0.9	−0.1	0.0	0.3
−0.6	−0.8	1.5	1.6	−0.1	−0.7	0.6	1.3
1.8	−0.2	1.6	−0.3	−0.8	1.5	1.0	−1.0
−1.3	−0.4	−0.3	−1.5	−0.5	1.7	1.1	−0.8
−2.6	1.6	−3.8	−1.8	1.9	1.2	−0.6	−0.4

Coefficente DC del blocco precedente : 12

Coefficenti quantizzati

15	0	−1	0	0	0	0	0
−2	−1	0	0	0	0	0	0
−1	−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	0	0	0	0	0	0	0

Sequenza codificata (RLE)

(2)(3), (1,2)(−2),
(0,1)(−1), (0,1)(−1),
(0,1)(−1), (2,1)(−1),
(0,0)

Tabella di Huffman

Sym	Code	Sym	Code
(2)	011	(3) (−2) (−1)	11 01 0
(0,0) (0,1) (1,2) (2,1)	1010 00 11011 11100		

Risultato : 31 bit
Originale : 64*8=512 bit

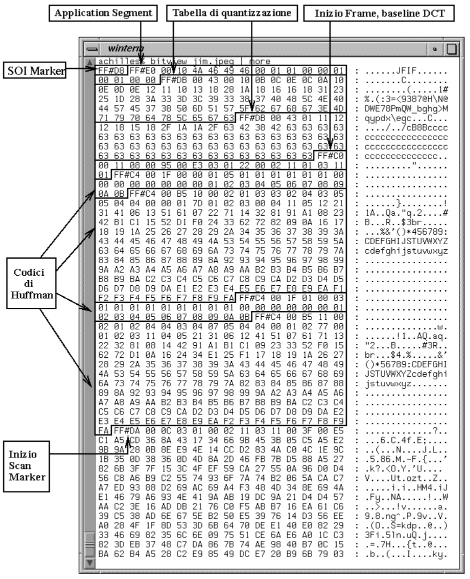
01111110110100000000001110001010

Sequenza di bit per il blocco 8x8

The JPEG Bitstream

JPEG image format

Type of segmen	Length of segment
Start of image	0
APP0	16
Quantisation table	67
Start of frame: baseline DCT	11
Huffman table	28
Huffman table	63
Start of scan	49363
End of image	0



```
achilles% bitview jim.jpeg | more
FF#D8 FF#E0 00 10 4A 46 49 46 00 01 01 00 00 01   : ......JFIF......
00 01 00 00 FF#DB 00 43 00 10 0B 0C 0E 0C 0A 10   : .......C........
0E 0D 0E 12 11 10 13 18 28 1A 18 16 16 18 31 23   : ........(.....1#
25 1D 28 3A 33 3D 3C 39 33 38 37 40 48 5C 4E 40   : %.(:3=<9387@H\N@
44 57 45 37 38 50 6D 51 57 5F 62 67 68 67 3E 4D   : DWE78PmQW_bghg>M
71 79 70 64 78 5C 65 67 63 FF#DB 00 43 01 11 12   : qypdx\egc...C...
12 18 15 18 2F 1A 1A 2F 63 42 38 42 63 63 63 63   : ..../../cB8Bcccc
63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63   : cccccccccccccccc
63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63   : cccccccccccccccc
63 63 63 63 63 63 63 63 63 63 63 63 63 63 FF#C0   : cccccccccccccc..
00 11 08 00 95 00 E3 03 01 22 00 02 11 01 03 11   : ........."......
01 FF#C4 00 1F 00 00 01 05 01 01 01 01 01 01 00   : ................
00 00 00 00 00 00 00 01 02 03 04 05 06 07 08 09   : ................
0A 0B FF#C4 00 B5 10 00 02 01 03 03 02 04 03 05   : ................
05 04 04 00 00 01 7D 01 02 03 00 04 11 05 12 21   : ......}........!
31 41 06 13 51 61 07 22 71 14 32 81 91 A1 08 23   : 1A..Qa."q.2....#
42 B1 C1 15 52 D1 F0 24 33 62 72 82 09 0A 16 17   : B...R..$3br.....
18 19 1A 25 26 27 28 29 2A 34 35 36 37 38 39 3A   : ...%&'()*456789:
43 44 45 46 47 48 49 4A 53 54 55 56 57 58 59 5A   : CDEFGHIJSTUVWXYZ
63 64 65 66 67 68 69 6A 73 74 75 76 77 78 79 7A   : cdefghijstuvwxyz
83 84 85 86 87 88 89 8A 92 93 94 95 96 97 98 99   : ................
9A A2 A3 A4 A5 A6 A7 A8 A9 AA B2 B3 B4 B5 B6 B7   : ................
B8 B9 BA C2 C3 C4 C5 C6 C7 C8 C9 CA D2 D3 D4 D5   : ................
D6 D7 D8 D9 DA E1 E2 E3 E4 E5 E6 E7 E8 E9 EA F1   : ................
F2 F3 F4 F5 F6 F7 F8 F9 FA FF#C4 00 1F 01 00 03   : ................
01 01 01 01 01 01 01 01 01 00 00 00 00 00 00 01   : ................
02 03 04 05 06 07 08 09 0A 0B FF#C4 00 B5 11 00   : ................
02 01 02 04 04 03 04 07 05 04 04 00 01 02 77 00   : ..............w.
01 02 03 11 04 05 21 31 06 12 41 51 07 61 71 13   : ......!1..AQ.aq.
22 32 81 08 14 42 91 A1 B1 C1 09 23 33 52 F0 15   : "2...B.....#3R..
62 72 D1 0A 16 24 34 E1 25 F1 17 18 19 1A 26 27   : br...$4.%.....&'
28 29 2A 35 36 37 38 39 3A 43 44 45 46 47 48 49   : ()*56789:CDEFGHI
4A 53 54 55 56 57 58 59 5A 63 64 65 66 67 68 69   : JSTUVWXYZcdefghi
6A 73 74 75 76 77 78 79 7A 82 83 84 85 86 87 88   : jstuvwxyz.......
89 8A 92 93 94 95 96 97 98 99 9A A2 A3 A4 A5 A6   : ................
A7 A8 A9 AA B2 B3 B4 B5 B6 B7 B8 B9 BA C2 C3 C4   : ................
C5 C6 C7 C8 C9 CA D2 D3 D4 D5 D6 D7 D8 D9 DA E2   : ................
E3 E4 E5 E6 E7 E8 E9 EA F2 F3 F4 F5 F6 F7 F8 F9   : ................
FA FF#DA 00 0C 03 01 00 02 11 03 11 00 3F 00 E5   : .............?..
C1 A5 CD 36 8A 43 17 34 66 9B 45 3B 05 C5 A5 E2   : ...6.C.4f.E;....
9B 9A 28 0B 8E E9 4E 14 CC D2 83 4A C0 4C 1E 9C   : ..(...N....J.L..
1B 35 0D 38 36 0D 4D 8A 2D 46 FB 7B D5 88 A5 27   : .5.86.M.-F.{...'
82 6B 3F 7F 15 3C 4F EF 59 CA 27 55 0A 96 D0 D4   : .k?.<O.Y.'U....
56 C8 A6 B9 C2 55 74 93 6F 7A 74 B2 06 5A CA C7   : V....Ut.ozt..Z..
A7 ED 93 88 D2 69 AC 69 A4 F3 48 4D 34 8E 69 4A   : .....i.i..HM4.iJ
E1 46 79 A6 93 4E 41 9A AB 19 DC 9A 21 D4 D4 57   : .Fy..NA.....!..W
AA C2 3E 16 AD DB 21 76 C0 F5 AB B7 16 EA 61 C6   : ..>...!v......a.
39 C5 38 AD 6E 67 5E B2 50 E5 39 76 14 D3 56 EE   : 9.8.ng^.P.9v..V.
A0 28 4F 1F 8D 53 3D 6B 64 70 DE E1 40 E0 82 29   : .(O..S=kdp..@..)
33 46 69 82 35 6C 6E 09 75 51 CE 6A E6 A0 1C C3   : 3Fi.5ln.uQ.j....
82 3D EB 37 48 C7 DA 86 7B 74 AE 98 40 B7 0C 15   : .=.7H...{t..@...
BA 62 B4 A5 28 C2 E9 85 49 DC E7 20 B9 6B 79 03   : .b..(...I....ky.
```

GIF, PNG, TIFF, JPEG standards at comparison

GIF – PNG

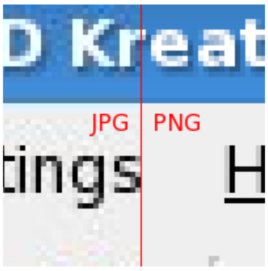
- Both GIF and PNG formats use lossless compression to achieve medium levels of compression on images
  - GIF works best on images with few colors or images in which one color is dominant. It acts best on identical, adjacent pixels (or rows of identical, adjacent pixels).
  - There is no loss during the compression process as long as the original image had fewer than 256 colors. The key to using GIFs effectively is to use the smallest possible number of colors. Also notable about GIFs is the fact that images can be transparent, animated, or both.
  - PNG is the other choice with higher number of colors.

JPEG

- The JPEG format uses lossy compression to achieve high levels of compression on images with many colors.
- The compression works best with continuous-tone images, that is, images where the change between adjacent pixels is small but not zero. JPEG images generally store 16 or 24 bits of color and thus are best for 16- or 24-bit images.
- Due to the noticeable loss of quality during the compression process, JPEGs should be used only where image file size is important, primarily on web pages.

- JPEG is a standard for digital photographs because it can save information on more than 16 million different hues. Its "lossy" compression has little effect on photographs. JPEG will produce a smaller file than PNG for photographic images (often 5–10 times) with negligible loss in quality.

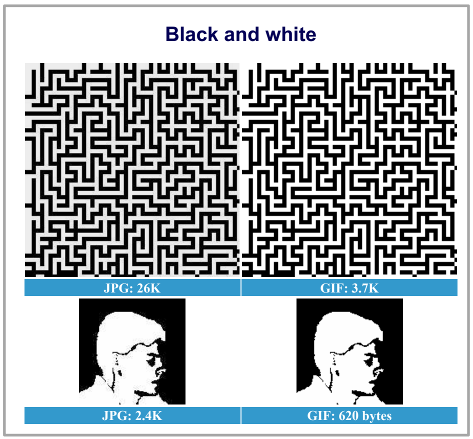
- PNG is a better choice than JPEG for storing images that contain text, line art, or other images with sharp transitions that do not transform well into the frequency domain.



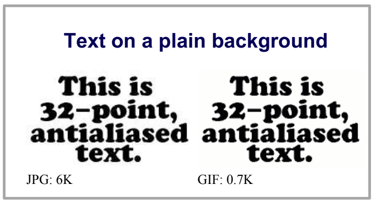
- Where an image contains both sharp transitions and photographic parts a choice must be made between the large but sharp PNG and a small JPEG with artifacts around sharp transitions.

Some general rules

- If your image is...
  - Black and white — Use GIF ■ sample
  - Text on a plain background — Use GIF ■ sample
  - Transparent or animated — Use GIF



50%

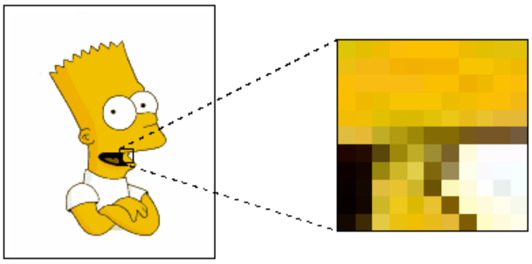

Text on a plain background
This is 32-point, antialiased text.
This is 32-point, antialiased text.
JPG: 6K
GIF: 0.7K

50%

– Computer-drawn line, cartoon art	Use GIF
– Small images, like icons, buttons	Use GIF
– Predominantly (>80%) one color	Consider GIF

Image size: 146 x 184, 75 colors.
File size:

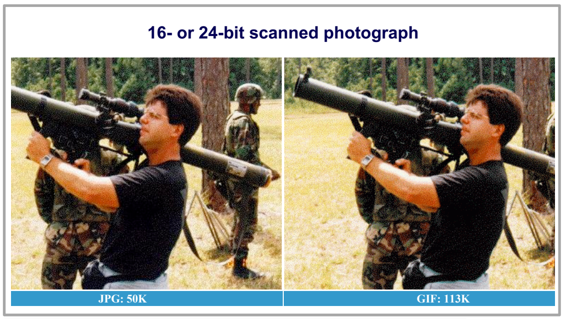
- 24 bpp: 80592 byts
- GIF (8 bpp): 3585 byte (FC=22.48)
- JPG (24 bpp): 4805 byte (FC=16.77)
- PPM.ZIP (24 bpp): 3698 byte (FC=21.79)



- If your image is...

– 16/24-bit scanned photograph	Use JPEG ■ sample
– Computer-drawn continuous-tone art	Use JPEG ■ sample
– Scanned images and photographs	Use JPEG
– (Large) images with a lot of detail	Use JPEG

50%



50%

Computer-drawn continuous-tone art
PLASMA
PLASMA
JPG: 7K
GIF: 19K

Image size: 244 x 334, 31322 colors
File size:

- 24 bpp: 244488 byte
- GIF (8 bpp): 49613 byte (FC=4.92)
- JPG (24 bpp): 16352 byte (FC=14.95)
- PPM.ZIP (24 bpp): 190977 byte (FC=1.28)

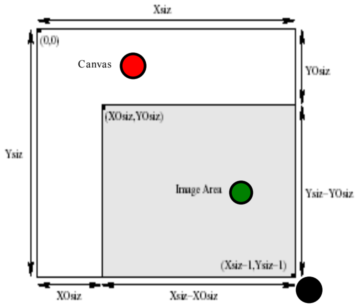
JPEG 2000 Standard

- JPEG 2000 is an ISO Standard (ISO/IEC 15444-1:2000) for images created by the Joint Photograph Expert Group committee in year 2000. Allows multi spectral imaging. JPEG2000 supports both lossy and lossless compression. JPEG2000 image files have the extension .jp2 or .j2f.
- JPEG2000 is not so widely used nor on the web as JPEG. It has no accepted way to embed EXIF data….

- Similarly to JPEG it requires that the image is converted from RGB into YCbCr or Y'CbCr to allow greater perceptual image quality for the same compression.

Reference Grid

- JPEG2000 supports multiple *image components* ● 1 to 255 or more. Each component can have different sizes and bit depths (1 to 32bits), and different alignments relative to each other
- JPEG2000 uses the concept of the *canvas* ● to align *multiple image components* in a single coordinate system. By default, image components are placed on the canvas so that the *image area* and canvas align ●
- Only those samples which fall within the image area actually belong to the image component. Thus the samples of component *i* are mapped as a rectangle having upper left hand sample with coordinates (*X0siz*, *Y0siz*) and lower right hand sample with coordinates (*Xsiz*-1, *Ysiz*-1)



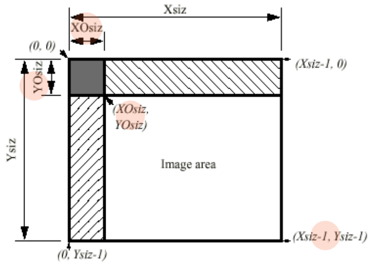


Image Tiling

- Each image component is further broken down into tiles. Tile sizes are variable, and can differ from component to component. Similar to blocks in JPEG, but more flexible.

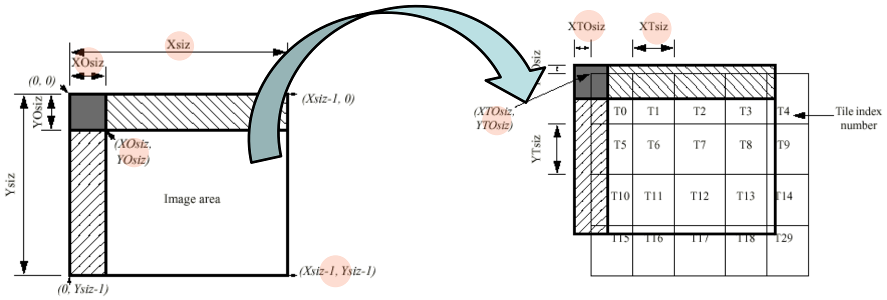


Image tiles relative to the reference

grid

The reference grid is partitioned into a regular sized rectangular array of tiles.
The *tile size* and *tiling offset* are defined on the reference grid, by dimensional pairs (XTsiz, YTsiz) and (XTOsiz, YTOsiz), respectively.

- By default, images will have one tile that has the same dimensions and offset on the canvas as the image.
- If the tile dimensions are smaller than the image dimensions and the tile offsets are different than the images offsets, some tiles may extend beyond the borders of the image.

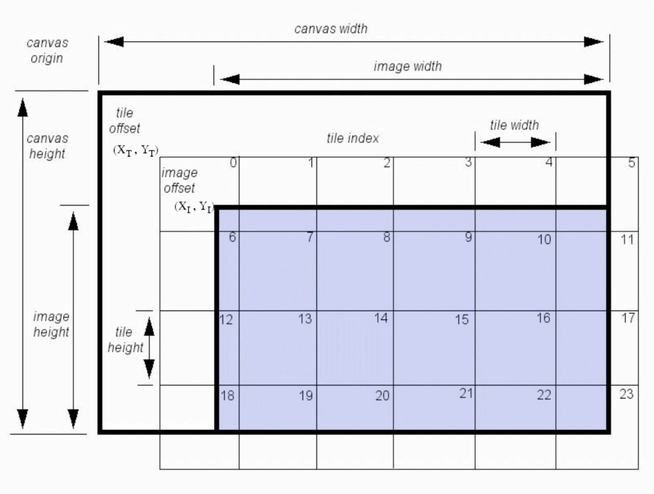
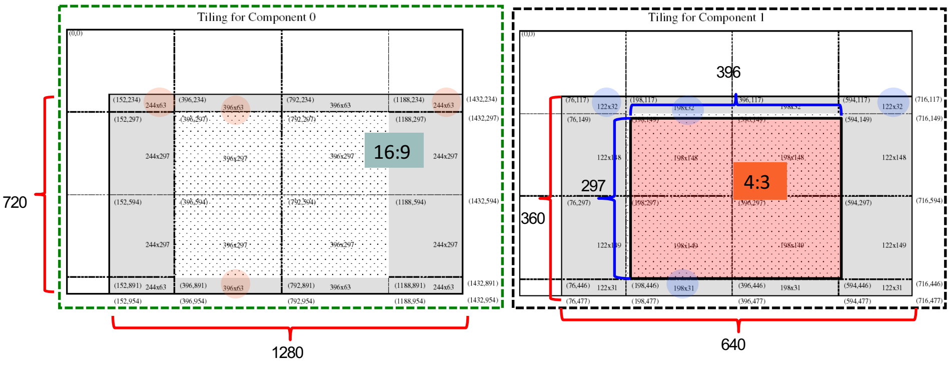


Image cropping and resizing

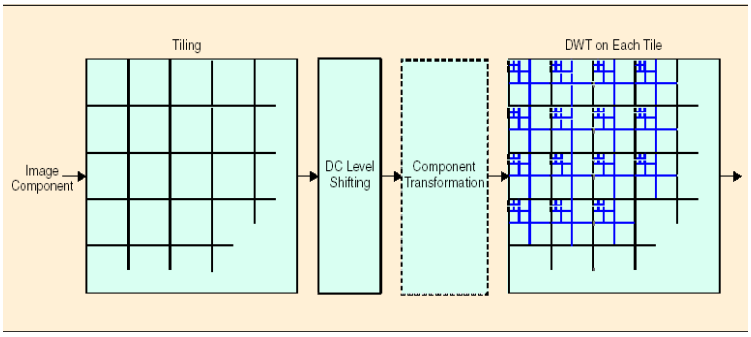


new image area

Here is an example that sub-samples an 1280 x 720 (16:9) image at 2:1 ratio on each side and then crops it to 4:3 aspect ratio. **New image size is 396x297**

Wavelet compression

- JPEG2000 uses Discrete Wavelet Transform in the lossy stage of image compression. Wavelet transform breaks down the image into multi resolution representations.
- For JPEG2000, the wavelet transform is applied to the image on a tile by tile basis.



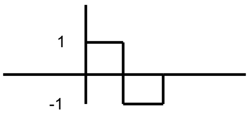
Discrete Wavelet Transform

- The first Discrete Wavelet Transform was invented by the mathematician Alfréd Haar:
  - For a list of 2*n* input numbers, the *Haar wavelet* transform simply pairs up input values, **storing the *difference*** and **passing the *sum.***
  - This process is repeated recursively, pairing up the sums to **provide the next scale**: finally resulting in 2*n* − 1 differences and one final sum.
- The Discrete Wavelet Transform has nice properties that make it an alternative to FFT:
  - it can be performed in *O*(*n*) operations
  - it captures not only some notion of the frequency content of the input (by examining it at different scales) but also captures the temporal content (i.e. the times at which the frequencies occur).

1D Discrete Wavelet Transform

- The Haar wavelet can be described as a step function:

$$F(x) \begin{cases} 1 & 0 <= x < \tfrac{1}{2} \\ -1 & \tfrac{1}{2} < \; x < =1 \\ 0 & \text{otherwise} \end{cases}$$



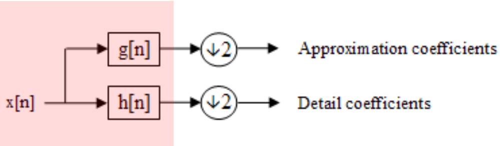
$$\text{2x2 matrix} \quad H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

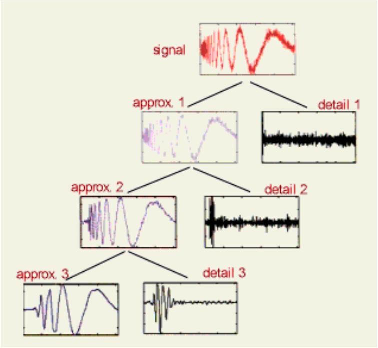
  - Given a sequence ($\mathbf{a_0, a_1, a_2, a_3 \ldots a_{2n+1}}$) of even lenght this can be transformed into a sequence of two-component vectors (($\mathbf{a_0,a_1}$),… ($\mathbf{a_{2n},a_{2n+1}}$)).
  - If one multiplies each vector with the matrix H one gets the result (($\mathbf{s_0,d_0}$)…..($\mathbf{s_n,d_n}$) of one stage of the Haar wavelet transform (***sum, difference***).
  - The two sequences s and d are separated and the process is repeated with the sequence ($\mathbf{s_0, s_1, s_2, s_3 \ldots s_{2n+1}}$)

- In the one dimensional Discrete Wavelet Transform case the signal is broken into subbands by passing it through a *low pass filter* and a *high pass filter*. The outputs give:
  - the approximation coefficients (from the *low-pass* filter)
  - the detail coefficients (from the *high-pass* filter)

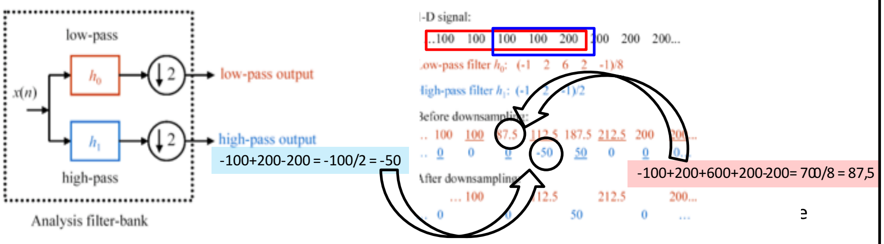
$$y_{\text{low}}[n] = \sum_{k=-\infty}^{\infty} x[k]g[2n-k]$$

$$y_{\text{high}}[n] = \sum_{k=-\infty}^{\infty} x[k]h[2n-k]$$





- Since half the frequencies of the signal have now been removed, half of the samples can be discarded according to Nyquist’s rule. The filter outputs are therefore downsampled by 2



-100+200-200 = -100/2 = -50

-100+200+600+200-200= 700/8 = 87,5

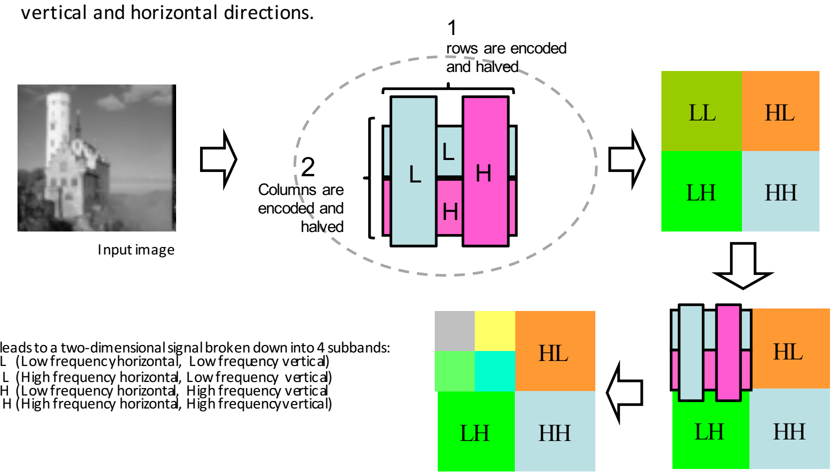
2D Discrete Wavelet Transform

- In the two-dimensional case, as in the 1D case, the signal is broken into subbands by passing it through *a low pass filter* and a *high pass filter*, and both subbands are downsampled by 2.
- According to the *Mallat method*, decomposition can be applied separably in the *vertical* and *horizontal* directions in the order

- After Discrete Wavelet decompostion has been performed, quantization matrix is applied to the decomposed image. JPEG2000 does not specify the use of particular quantization matrices.
- Uniform quantization is performed within each subband, with different levels of quantization for each subband. Generally, the higher frequency subbands are quantized more coarsely, since humans have lower contrast sensitivity to high frequency information.

Mallat method

- 1D Discrete Wavelet Transform is applied alternatively to vertical and horizontal direction line by line. Decomposition is iteratively applied.
- The LL band is recursively decomposed, first vertically and then horizontally. Since downsampling is performed at each pass, at each iteration the image halves its size in the vertical and horizontal directions.



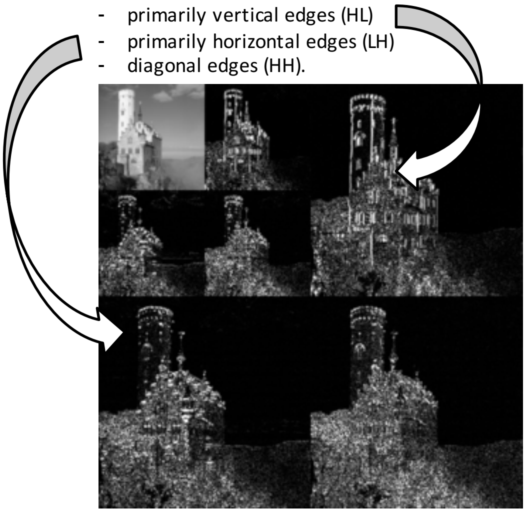
This leads to a two-dimensional signal broken down into 4 subbands:

- LL (Low frequency horizontal, Low frequency vertical)
- HL (High frequency horizontal, Low frequency vertical)
- LH (Low frequency horizontal, High frequency vertical
- HH (High frequency horizontal, High frequency vertical)

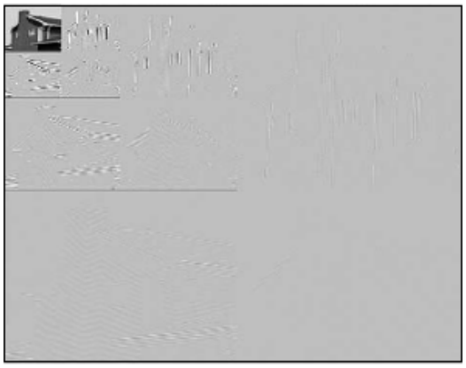
2D Wavelet Decomposition

•Conceptually, for a particular image, these subbands translate to:

- low-frequency approximation of the original (LL)
- primarily vertical edges (HL)
- primarily horizontal edges (LH)
- diagonal edges (HH).

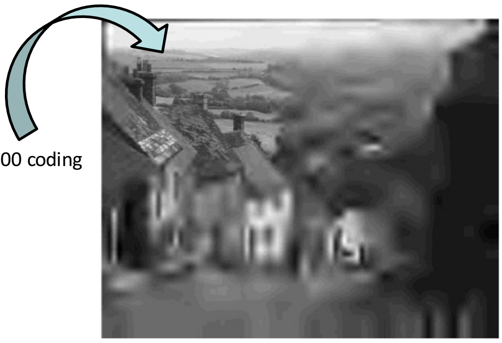


An example



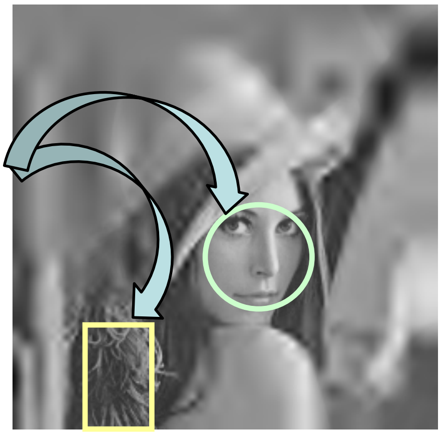
Region of Interest (ROI) coding

- JPEG 2000 offers increased flexibility that can make it more applicable than JPEG, and has other interesting feature like ROI coding and progressive transmission
- In ROI coding, portions of an image are stored at higher quality than the rest of the image. This is useful, because we may care more about detail in some portions of an image than in others.



An example of
Region of Interest JPEG2000 coding

- ROI is easy to do when the image is stored compressed in a multi resolution format.
  - start with a ROI mask, which marks out a region of the image to store at higher quality.
  - the wavelet coefficients corresponding to the transform of the mask have to be stored at higher quality (quantized less coarsely).
  - apply the transform to the mask, and look at which coefficients fall in the mask.



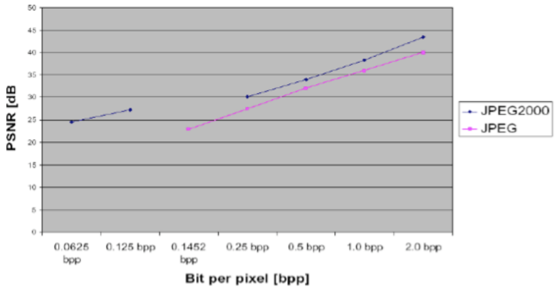
Coefficients here are quantized less coarsely at any subband

JPEG2000 vs JPEG

- With respect to JPEG it allows space saving in the order of 20%-30%. Therefore it appears to be particularly suited for large images.

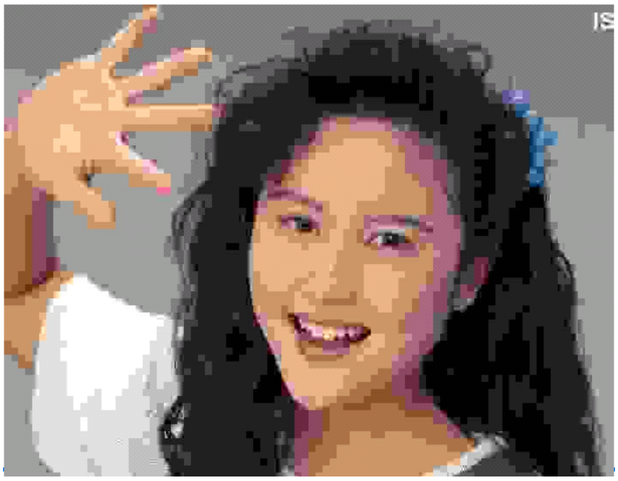
- However this is not the primary motivation for its use. More important JPEG 2000 employs multiresolution and can arrange a large range of bit rates (both very low and very high compression rates are supported). With JPEG if we want to trasmit over low bit rate we should first reduce the resolution and then encode.

- The wavelet representations of an image generally perform better than DCT representations for lossy image compression, as there is less perceptual loss for the same bit rate even when performed on the same block size.



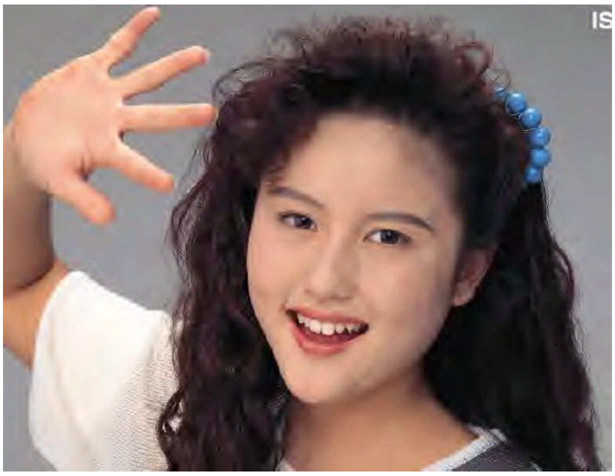
- Multi-resolution wavelet representations give better performance because:
  - Multi-resolution representations are more similar to how the human visual system represents images. Quantization matrices can be chosen that more closely match and exploit the characteristics of the human visual system
  - The wavelet basis functions are smoother than the DCT basis functions (which tend to be blocky), and are more natural and pleasing to the eye.

A comparative example



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)