

# ELEC S347F Multimedia Technologies

A decorative graphic consisting of three horizontal lines with a circuit-like pattern on the left side, featuring three small circles and connecting lines.

## Image Representation and Compression



# Outline



- Color Basics and Color Models
- Image Representation
  - Bitmap Graphics
  - Color Lookup Table
  - Vector Graphics
- Image Format and Compression
  - GIF
  - Lossless JPEG
  - Lossy JPEG

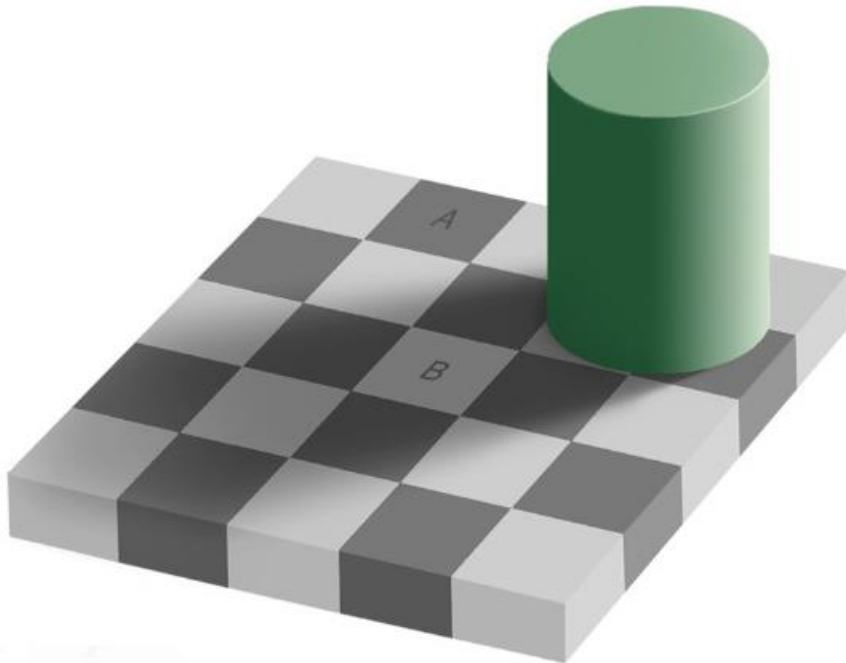
# What are the Colors of the Dress?

■ Gold and White or Black and Blue?



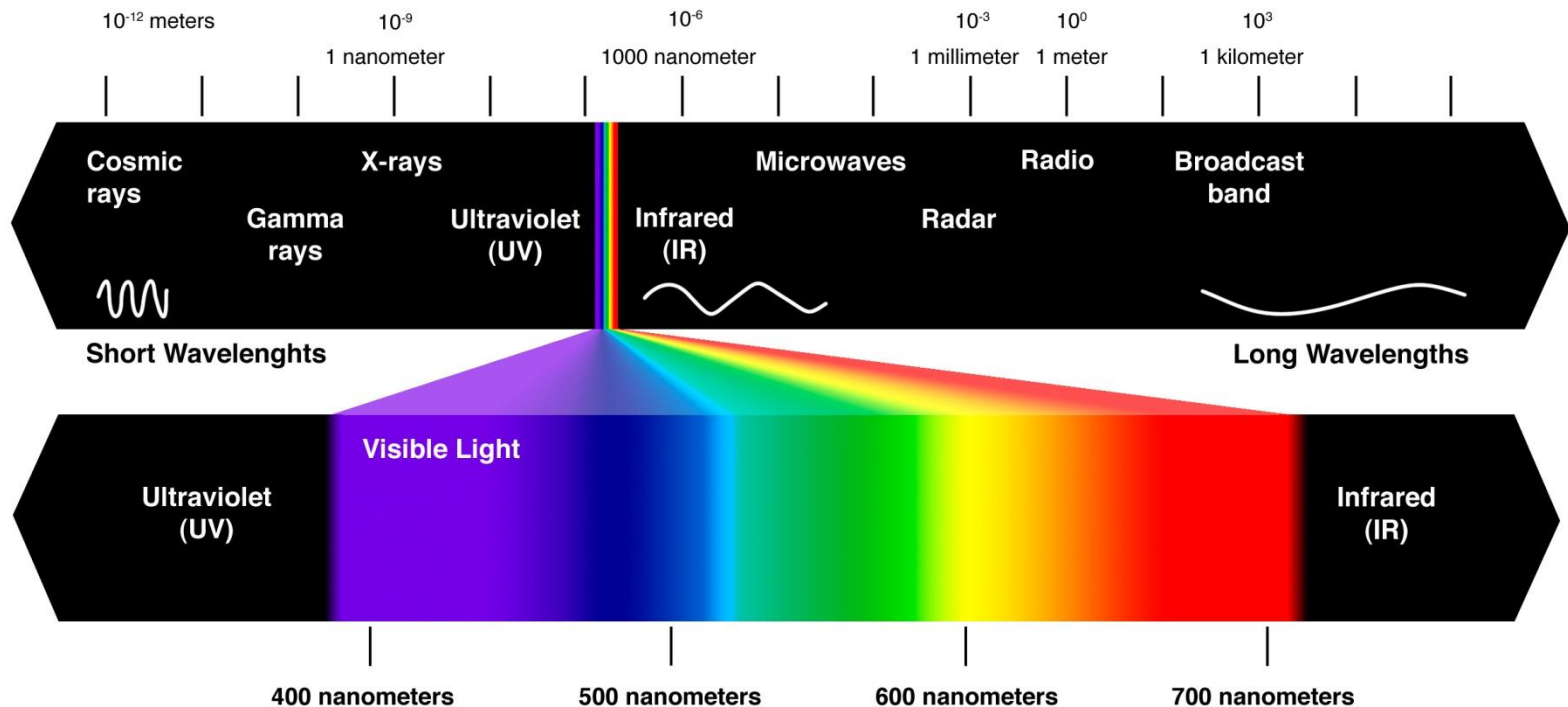
# Which One is Darker?

■ A or B?



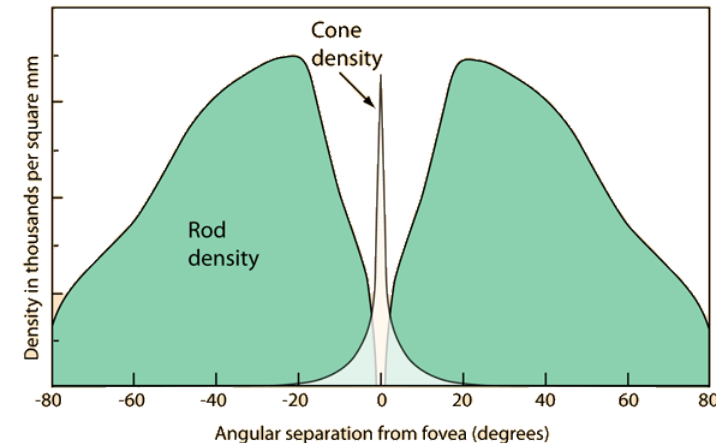
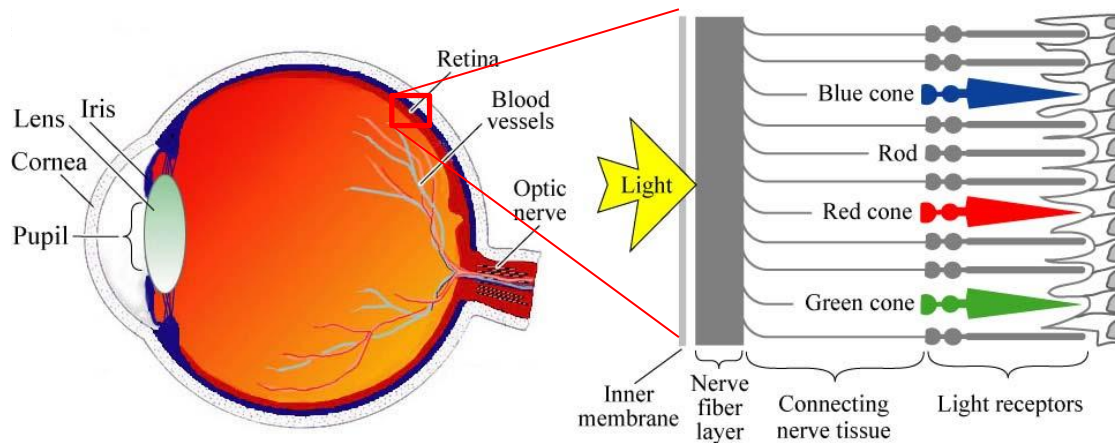
# Color Basics

- Visible light is an electromagnetic wave in the 400 nm to 700 nm range



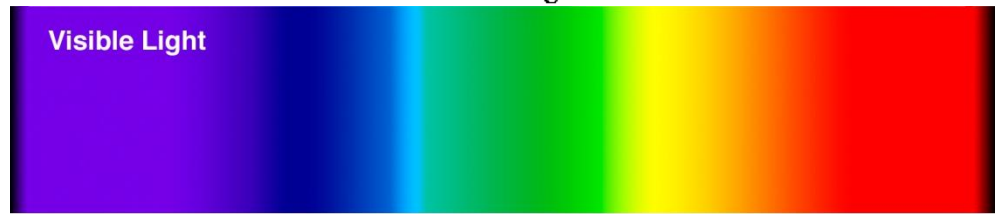
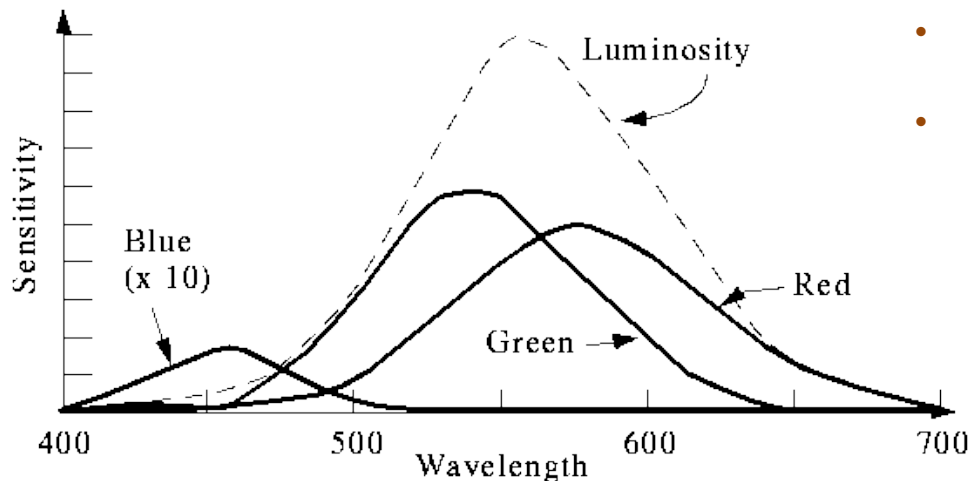
# Retina: Rods and Cones

- Human eyes function on the same principle as a camera
  - Retina has two types of sensors: rod and cone
  - The rods are sensitive to magnitude (i.e. sensitive to light intensities only)
  - The cones are sensitive to wavelengths (i.e. sensitive to color only)



# Cones and Color Composition

The spectral-response function of the cones and the luminous-efficiency function of the retina



400 nanometers

500 nanometers

600 nanometers

700 nanometers

There are 3 types of cones: S type, M type and L type. Each type of cones responds differently to various frequencies of light.

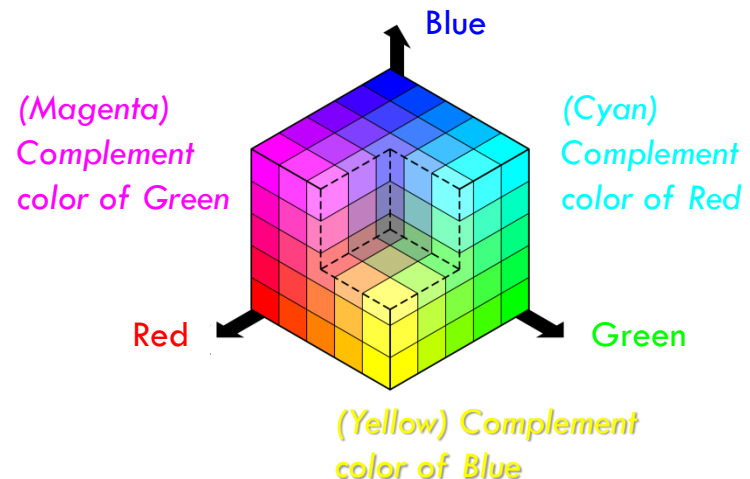
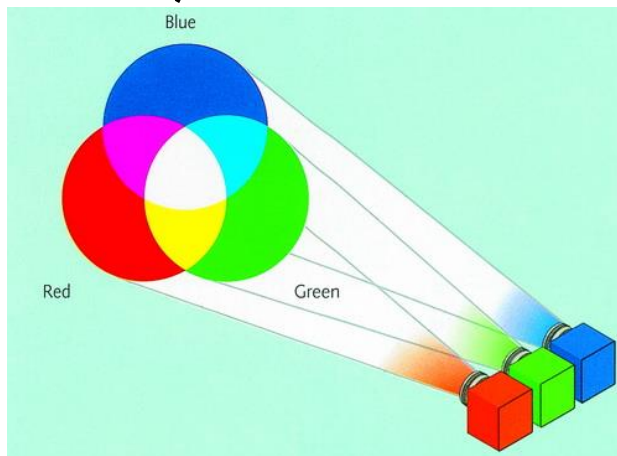
- S type is more sensitive to a short-wavelength light (Peak around 420 nm to 440 nm)
- M type is more sensitive to medium-wavelength light (Peak around 534 to 545 nm)
- L type is more sensitive to long-wavelength light (Peak around 564 to 580 nm)

A color is the perceived sum of S-, M-, L-cones

- For yellow color, it is perceived when the L cones are stimulated slightly more than the M cones
- For red color, it is perceived when the L cones are stimulated significantly more than the M cones
- For violet color, it is perceived when the S receptor is stimulated more than the other two

# Color Model: RGB Additive Model

- A color can be represented as the weighted sum of **Red**, **Green** and **Blue** primitive colors
  - $Color = (R, G, B)$
  - where  $0 \leq R, G, B \leq 1$ , the weighting of the primitive colors
  - This is called the RGB Additive Model
  - Typically used in light source and displaying devices (e.g. monitor)





# Color Model: CMY Subtractive Model

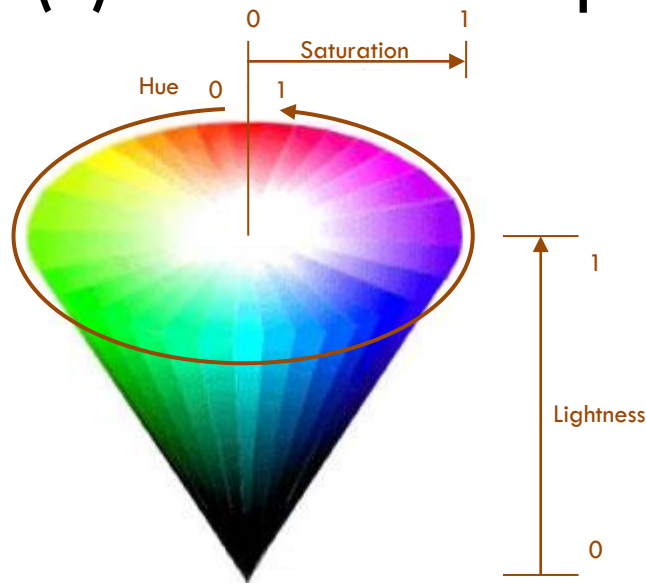
- Another model is the CMY Subtractive Model
  - Cyan (C), Magenta (M) and Yellow (Y) are complementary colors of RGB respectively
  - Typically used in printing
  - Unlike displaying devices which actively output certain combination of wavelength to display a color,
  - Printed materials passively reflect the color from a light source
  - e.g. the yellow pigments on a paper absorb the blue light
  - (i.e. no blue light reflected from yellow ink, so you can see the reflected yellow light)

# Color Model: CMYK Model

- In practice, the CMYK model is used more often than the CMY model
  - In theory CMY combined together creates a perfect black
  - However, in practice black cannot be easily produced via combination of CMY due to imperfect nature of the paint
  - Therefore, black (K) ink is used instead and this form the CMYK model
- Conversion between RGB and CMYK models
  - $C' = 1 - R, M' = 1 - G, Y' = 1 - B$
  - $K = \min(C', M', Y')$
  - $(C, M, Y, K) = (C' - K, M' - K, Y' - K, K)$

# Color Model: HSL Model

- Hue (H) refers to pure color
- Saturation (S) refers to the perceived intensity of a color
- Lightness (L) refers to the perceived brightness



# Image Representation

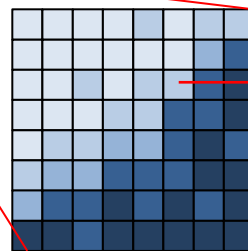


# Digital Image

- Digital image is composed by small dots called pixels
  - For each pixel, it represents a color
  - The storage size of each pixels is called color depth
  - Typical color depths are 1, 8, 16, 24, 32 bits
  - The larger the color depth, the more colors can be represented
- The color information are stored as the same 2D orientation of the pixels



Represented as  
bitmap



Pixel

B9CDE5

The pixel value can be represented by RGB, CMY, HSL, or other color models.

In this example, it is using 24-bit RGB model where  $R' = 89_{16}$ ,  $G' = CD_{16}$ ,  $B' = E5_{16}$ .  
 $R = R'/FF_{16}$ ,  $G = G'/FF_{16}$ ,  $B = B'/FF_{16}$ .  
Here, 24 bits refer to the color depth of a pixel.

# Monochrome Image

## ■ For a HD image

- The image width is 1920 (contains 1920 dots)
- The image height is 1080 (contains 1080 dots)
- The resolution is 1920 x 1080
- The aspect ratio is  $1920:1080 = 16:9$
- The raw image size = resolution x color depth

## ■ For monochrome image,

- Color depth = 1 bit
- Each pixel represents either a black dot or white dot
- For a HD monochrome image, it's raw image size =  $1920 \times 1080 \times 1 \text{ bit} = 259,200 \text{ bytes}$

# Grayscale Image

- For grayscale image,
  - The color depth = 8 bits (1 byte)
  - Each pixel can represent  $2^8 = 256$  different levels of gray colors



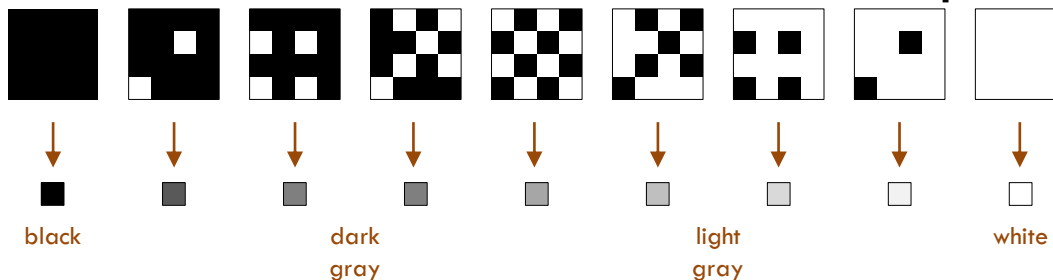
0

255

- i.e. value 0 = pure black, value 255 = pure white, value between 1~254 = from dark gray to light gray
- For a HD grayscale image, it's raw image size =  $1920 \times 1080 \times 8 \text{ bits} = 2,073,600 \text{ bytes}$

# Dithering

- How to represent the different gray level?
  - For traditional monitors and printers, there are monochrome (i.e. black and white, on and off only, no gray color)
  - Dithering: use a larger and different pattern of monochrome pixels to approximate the greyscale levels of the input image
  - By varying the pattern of black and white pixels, different color darkness can be represented





# Color Depth

## ■ 8-bit Color

- The color depth = 8 bits (1 byte) where G and R = 3 bits, B = 2 bits
- It can represent  $2^8 = 256$  distinct colors
- Why? Humans are more sensitive the red and green than to blue, by a factor of approximately 1.5 times

## ■ For high color image,

- The color depth = 16 bit (2 bytes) where G = 6 bits, R and B = 5 bits
- There can represent  $2^{16} = 65536$  distinct colors
- Alternative design: G = R = B = 5 bits, Alpha (transparency) = 1 bit

## ■ For true color image,

- The color depth = 24 bits (3 bytes) where G = R = B = 8 bits
- There can represent  $2^{24} = 16,777,216$  distinct colors

## ■ For 32 bits color depth images,

- The extra 8-bit represents the degree of transparency (opacity) of a pixel
- From 0 (non-transparent, i.e. 0%) to 255 (full transparent, i.e. 100%)

# True Color Image

- For a HD true color image
  - Resolution = 1920 x 1080; Color Depth = 24 bits
  - Raw image size = 1920 x 1080 x 24 bits = 6,220,800 bytes
- For a HD image with 32-bit color depth
  - Raw image size = 1920 x 1080 x 24 bits = 8,294,400 bytes
- As you can see, if the images are not compressed, their raw sizes are too large for delivery, storage and manipulate
  - How? There are a lot of redundant information exists in images

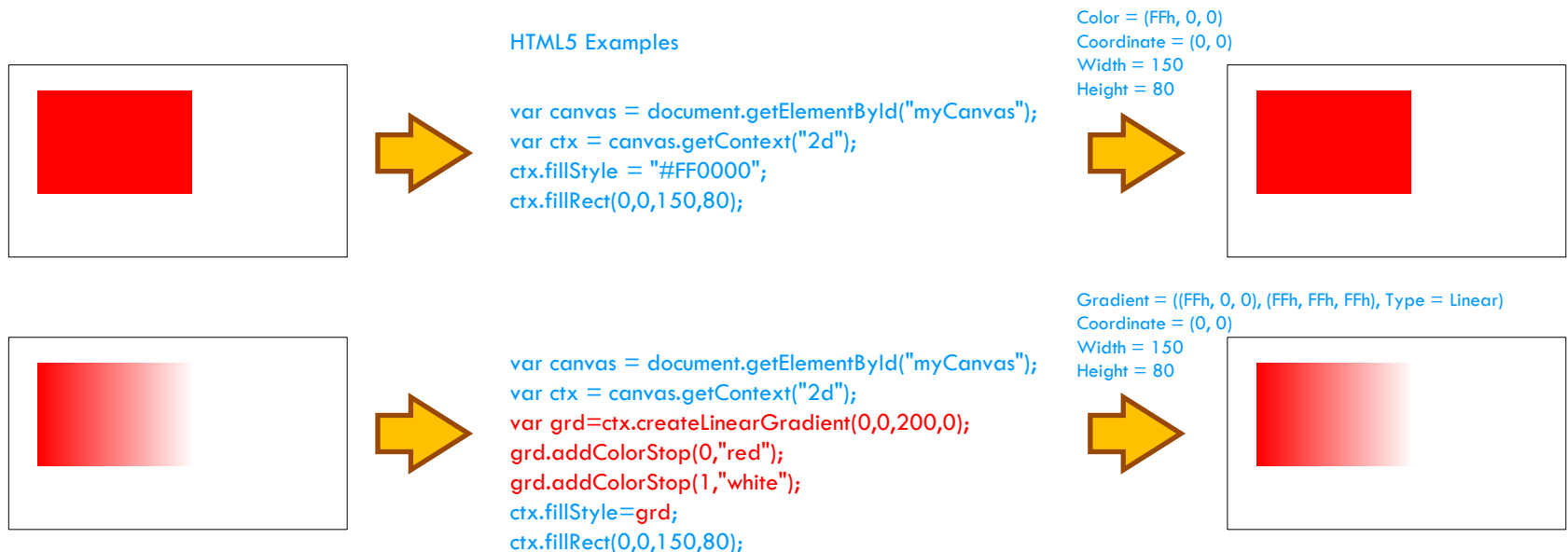
# Image Size Reduction



- There are different ways to reduce the size of an image
  - Vector Graphics
  - Color Lookup Table
  - Lossless Compression
  - Lossy Compression

# Vector Graphics

- Images, especially computer-generated, may be represented using high level descriptions
  - Examples: HTML5, SVG, Adobe Illustrator
  - The reconstruction process is called rendering



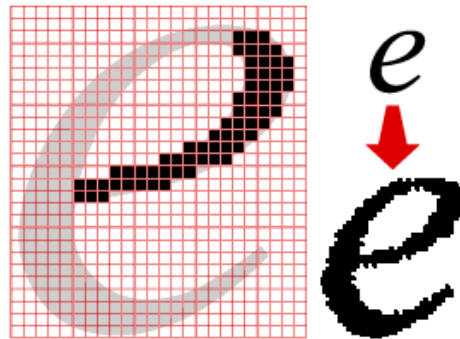
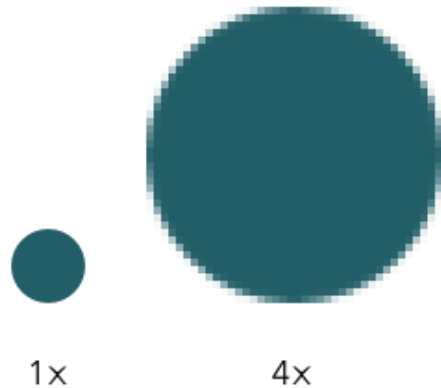
# Bitmap vs. Vector Graphics

- The image size of vector representation is independent of its resolution
  - e.g. for a 100x100 image containing a circle and another 10,000 x 10,000 image containing a circle
  - Both have the same self-information (require the same number of bits for storing the radius, origin and color)
  - For bitmap representation, 100x100 image has  $10^4$  pixels and 10,000x10,000 image has  $10^8$  pixels
  - The later one definitely require more space in its representation
- No blurriness even if the image is enlarged
  - Since multiple resolutions can be rendered by the vector format
- However, not all images can be represented in vector form easily
  - Good for primitive shapes but not camera taken photos

# Bitmap vs. Vector Graphics

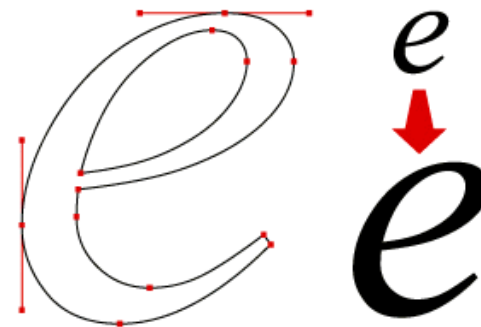
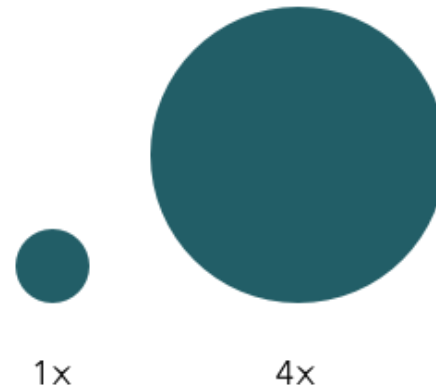
## Bitmap (Raster)

■ JPEG, GIF, PNG



## Vector

■ AI, SVG



# Color Lookup Table

- If an image consists of a few colors only, it may be good to store the color index for each pixel instead of the color itself
- Example: Graphics Interchange Format (GIF) supports image of up to 256 24-bit colors

Each pixel store a true color value

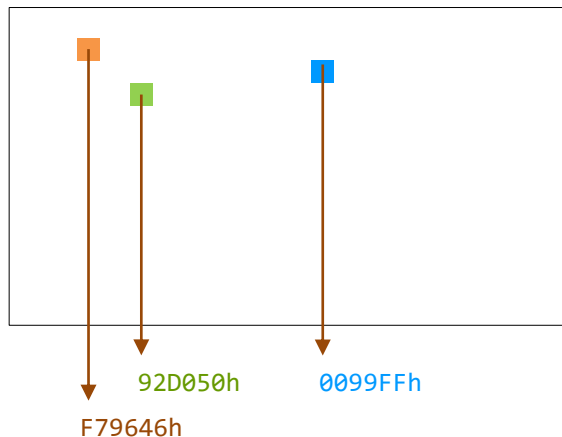


Image size = width x height x 24 bits

Each pixel store an index value

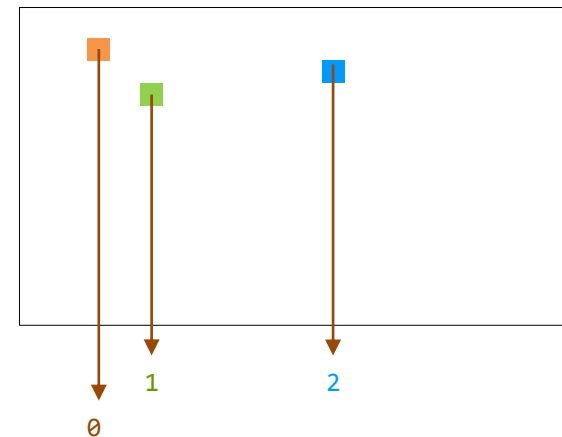


Image size = width x height x 8 bits + 256 x 24 bits

Color Lookup Table (Palate)

0	F79646h
1	92D050h
2	0099FFh
...	...
255	

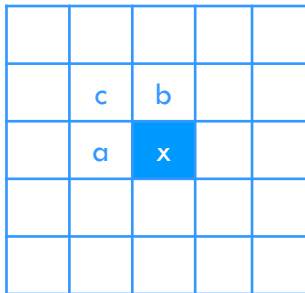
# GIF

- The image pixel data (i.e. color index) are scanned horizontally from top left
  - The sequence of pixel data are then encoded using LZW coding
  - The compression rate will be dependent on the content of the image
- Different lossless compression schemes
  - PCX: Run-length Encoding (RLE)
  - GIF: Dictionary-based Encoding (LZW)
  - PNG: Prediction-based Compression
  - Lossless JPEG: Prediction-based Compression



# Lossless JPEG

- Lossless JPEG uses a predictive scheme for modeling and applies entropy coding on the residuals
  - There are 8 different predictive schemes from which the user can select
  - If compression is performed in a non-real time environment, all eight modes of prediction can be tried, and the one that gives the most compression is used
  - The mode used to perform the prediction can be stored in a 3-bit header along with the compressed file



Scheme	Prediction		Remarks
0	-	No prediction	Always used for the first sample of the first line
1	a	1D Predictor	Always used for the first line of samples
2	b	1D Predictor	Always use for the first column of samples
3	c	1D Predictor	
4	$a+b-c$	2D Predictor	
5	$a+(b-c)/2$	2D Predictor	
6	$b+(a-c)/2$	2D Predictor	
7	$(a+b)/2$	2D Predictor	

# Why Lossy Compression?



- Compression ratio of lossless methods is not high enough for multimedia data
- By cleverly making a small sacrifice in terms of fidelity of data
- A higher compression ratios could be achieved
- Sacrifice information that is psycho-physically unimportant

# Exploiting Psycho-Visual Redundancy

- Human perception is more sensitive to brightness than chrominance (color)
  - Separate the brightness information from the color and process them independently
- More sensitive to differences between dark intensities than bright ones
  - Encode  $\log(\text{intensity})$  instead of intensity
- More sensitive to high spatial frequencies of green than red or blue
  - Sample green at highest spatial frequency, blue at lowest
- More sensitive to differences of intensity in green than red or blue
  - Use variable quantization: devote most bits to green, fewest to blue
- Humans are more immune to loss of higher spatial frequency components than loss of lower frequency components
  - A large majority of useful image contents change relatively slowly across pixels
  - Represent the image using spatial frequency and remove those high spatial frequency components

# Joint Photographic Experts Group (JPEG) Standard

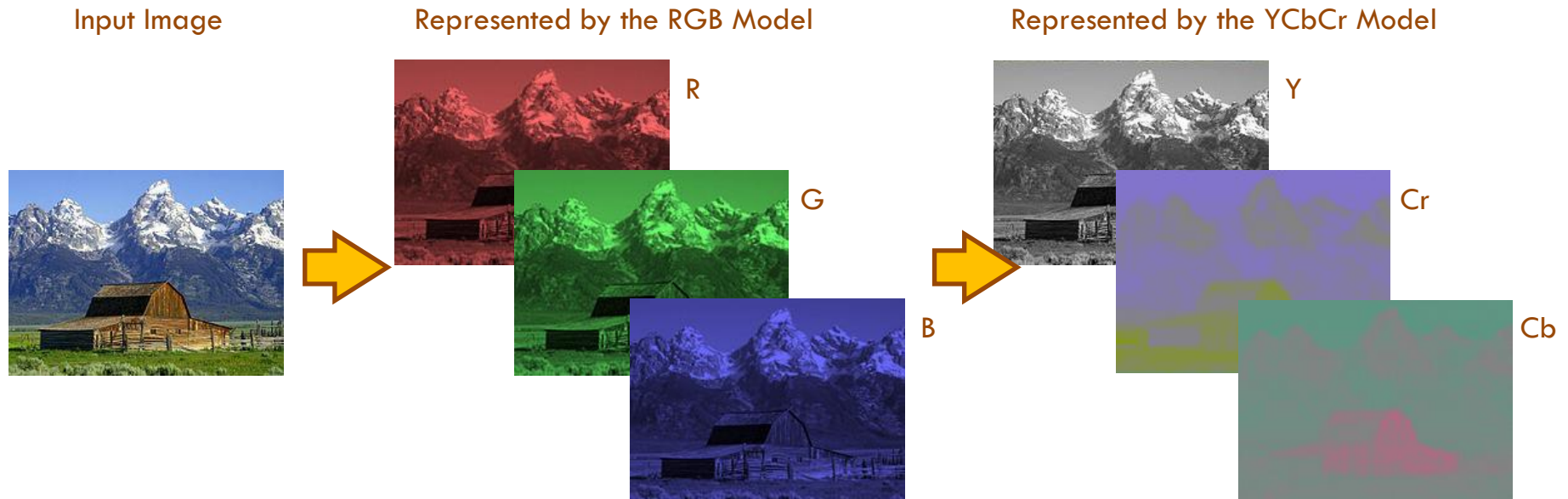


# JPEG: Color Model

- How to separate the brightness information from color?
- The YUV color model
  - Y is the luminance
  - i.e. brightness of the image, grayscale of the color
    - $Y = 0.299R + 0.587G + 0.114B$
  - Chrominance is defined as the difference between a color and a reference white at the same luminance
    - $U = B - Y$
    - $V = R - Y$
- The YCbCr color model
  - It is scaled and shifted YUV model
  - Same Y definition as YUV's Y
  - $Cb = (B - Y)/2 + 0.5$  (Cb is always in the range of 0 to 1)
  - $Cr = (R - Y)/1.6 + 0.5$  (Cr is always in the range of 0 to 1)

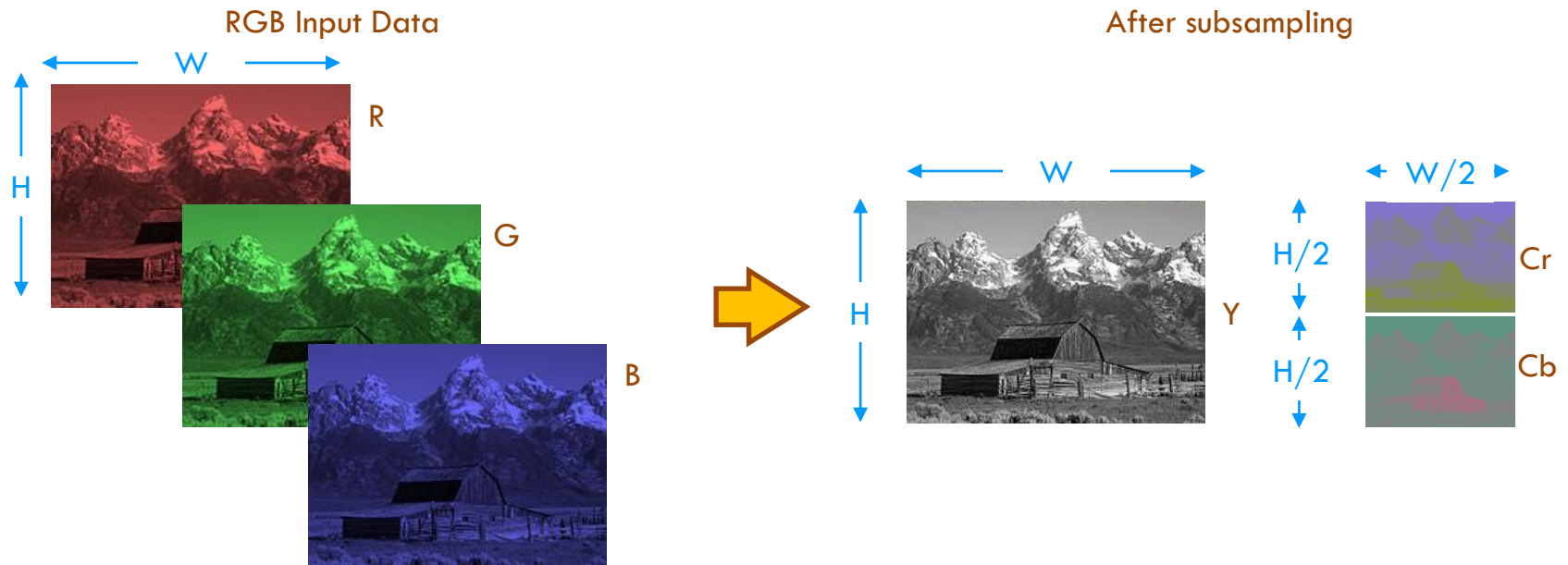
# JPEG: Color Model

- JPEG algorithm first transforms RGB into the YCbCr color space



# JPEG: Color Model

- Human perception is more sensitive to brightness
  - Therefore, luminous (Y component) is more important than the chrominance (Cb, Cr values)
  - Subsample the chrominance components using 4:2:0 (i.e.  $\frac{1}{2}$  horizontal resolution,  $\frac{1}{2}$  vertical resolution)



# JPEG: Transformation

- A large majority of useful image contents change relatively slowly across pixels
  - Do not store the intensities and chrominance values directly, but the changes between relative pixels
- Transform the pixel values from spatial domain into frequency domain
  - Then can isolate the frequency changes between relative pixels
  - JPEG algorithm divides the Y, Cb, Cr components into 8 pixels x 8 pixels blocks
  - If the no. of pixels of width and height is not a multiple of 8, add empty pixels around the edges
  - For each block, take the 2D discrete cosine transform (DCT)



# Discrete Cosine Transform

## ■ Forward 2-Dimensional DCT

$$F(u, v) = \frac{2}{\sqrt{nm}} C(u) C(v) \sum_{y=0}^{m-1} \sum_{x=0}^{n-1} I(x, y) \cos\left(\frac{(2x+1)u\pi}{2n}\right) \cos\left(\frac{(2y+1)v\pi}{2m}\right)$$

■ where  $I(x, y)$  is the pixel value,  $C(i) = \begin{cases} \frac{1}{\sqrt{2}}, & i = 0 \\ 1, & i \neq 0 \end{cases}$

■ For the JPEG algorithm,  $n = m = 8$  (8 x 8 pixel<sup>2</sup> block)

## ■ Inverse 2-Dimensional DCT

$$I(y, x) = \frac{2}{\sqrt{nm}} \sum_{y=0}^{m-1} \sum_{x=0}^{n-1} F(v, u) C(u) C(v) \cos\left(\frac{(2x+1)u\pi}{2n}\right) \cos\left(\frac{(2y+1)v\pi}{2m}\right)$$

# Understanding DCT

## ■ Forward 1-Dimensional DCT

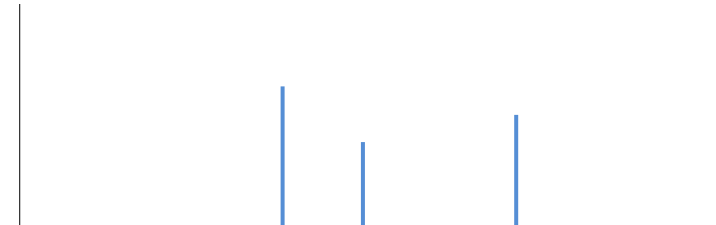
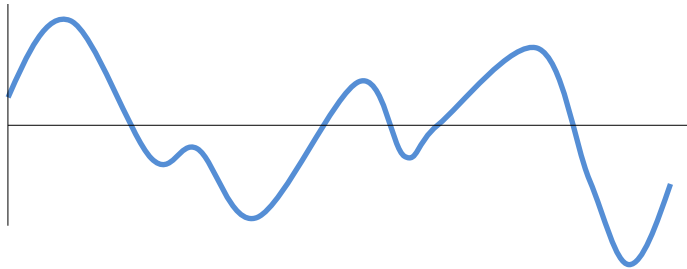
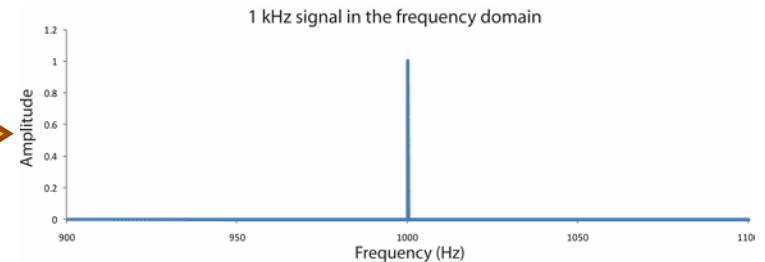
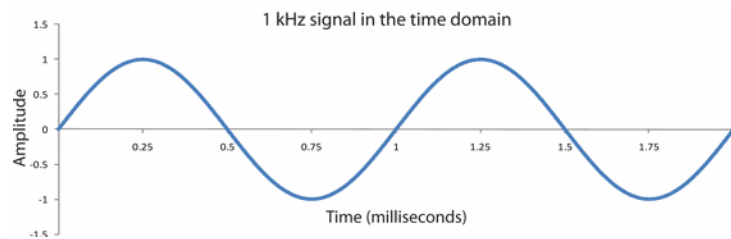
$$■ F(u) = \frac{\sqrt{2}}{\sqrt{n}} C(u) \sum_{x=0}^{n-1} I(x) \cos\left(\frac{(2x+1)u\pi}{2n}\right)$$

## ■ Inverse 1-Dimensional DCT

$$■ I(x) = \frac{\sqrt{2}}{\sqrt{n}} \sum_{u=0}^{n-1} F(u) C(u) \cos\left(\frac{(2x+1)u\pi}{2n}\right)$$

# Understanding DCT

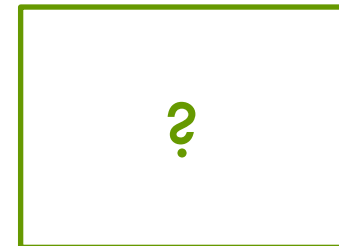
## ■ For audio (1-dimensional)



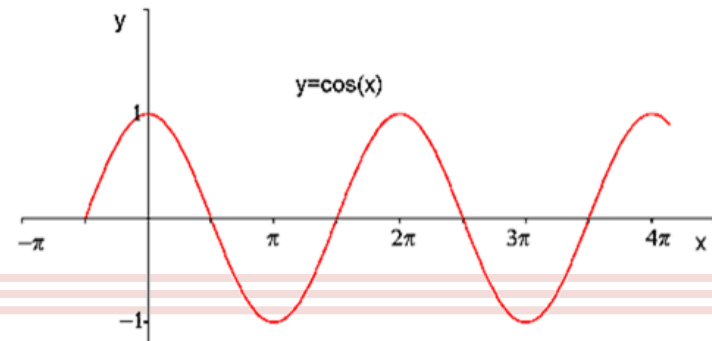
## ■ For image (2-dimensional)



255	123	045	019	039	084
094	187	081	029	145	181
227	084	089	184	189	017
045	082	175	089	215	089
207	145	192	007	008	157
147	035	089	178	189	163
046	084	178	045	089	168
049	179	103	100	048	235



# Understanding DCT



■ Consider the below sequences of integers

■  $I(x) = \{1, 0, -1, 0, 1, 0, -1, 0\}$

■ It can be represented as a cosine function

■  $I(x) = \cos\left(\frac{\pi}{2} \cdot x\right)$

■  $I(x) = \{5, 0, -5, 0, 5, 0, -5, 0\}$

■ It can be represented as  $I(x) = 5\cos\left(\frac{\pi}{2} \cdot x\right)$

■  $I(x) = \{+5, -5, +5, -5, +5, -5, +5, -5\}$

■ It can be represented as  $I(x) = 5\cos(\pi \cdot x)$

■ How about  $I(x) = \{15, -5, -5, -5, 15, -5, -5, -5\}$ ?

■ It can be represented as  $I(x) = 10\cos\left(\frac{\pi}{2} \cdot x\right) + 5\cos(\pi \cdot x)$

# Understanding DCT

- An integer sequence can be considered as the summation of some cosine functions with various frequencies

- $I(x) = \frac{\sqrt{2}}{\sqrt{n}} \sum_{x=0}^{n-1} F(u)C(u) \cos\left(\frac{(2x+1)u\pi}{2n}\right)$

- where  $F(u)$  is the coefficient of the  $u^{\text{th}}$  cos. function

- Actually, an image block can be considered as a 2-dimensional integer sequence

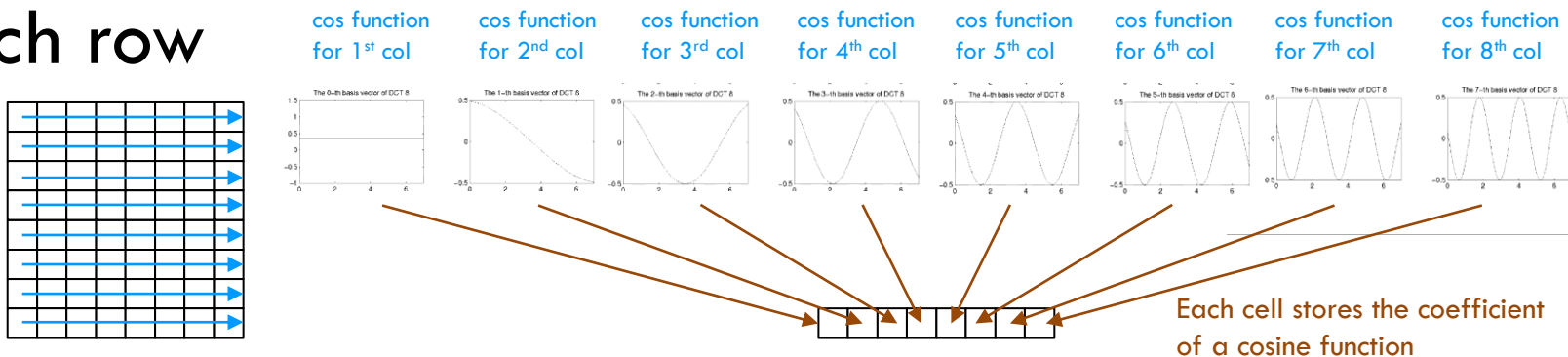
- Can be represented by the summation of some 2D cos. functions

- i.e. summation of the summation of some cos. functions

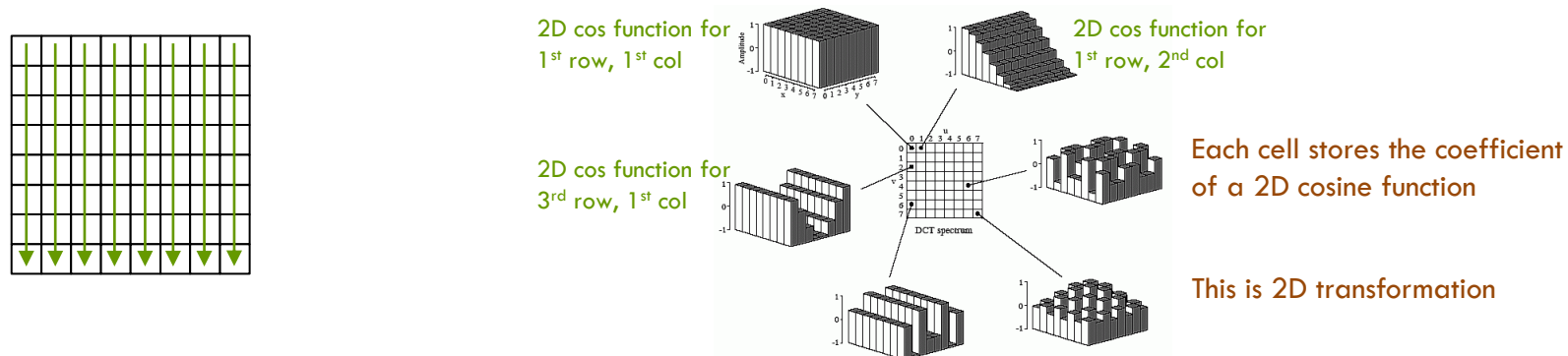
$$I(y, x) = \frac{2}{\sqrt{nm}} \sum_{y=0}^{m-1} \sum_{x=0}^{n-1} F(v, u)C(u)C(v) \cos\left(\frac{(2x+1)u\pi}{2n}\right) \cos\left(\frac{(2y+1)v\pi}{2m}\right)$$

# Understanding DCT

- First perform 1D transformation for pixels on each row

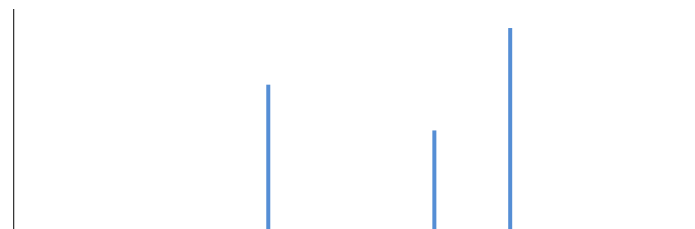
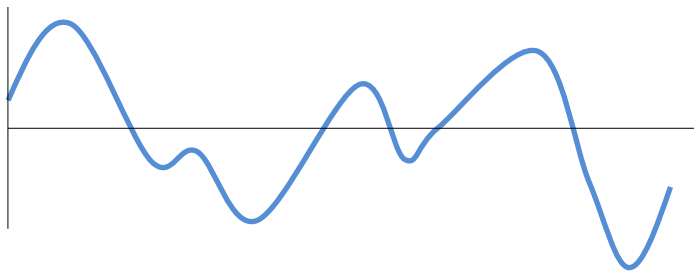
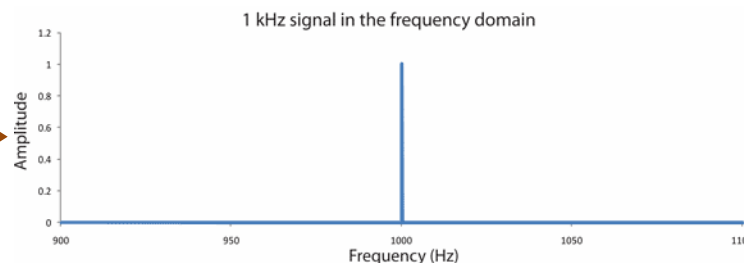
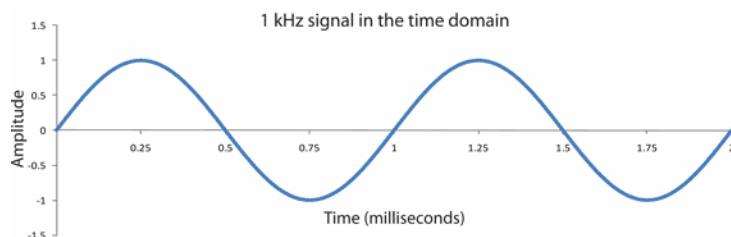


- Then on top of the 1D result, perform the same transformation on each column



# Spatial to Frequency Domain

## ■ For audio (1-dimensional)

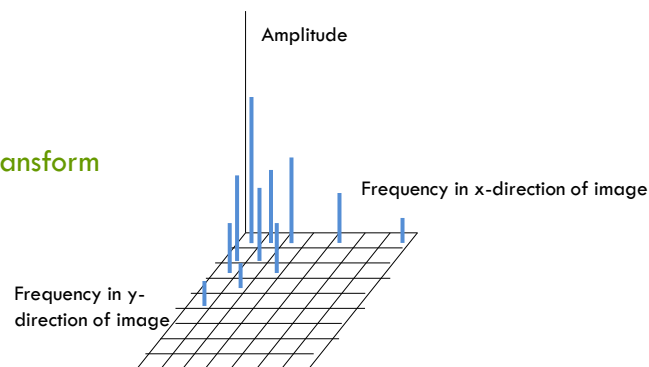


## ■ For image (2-dimensional)



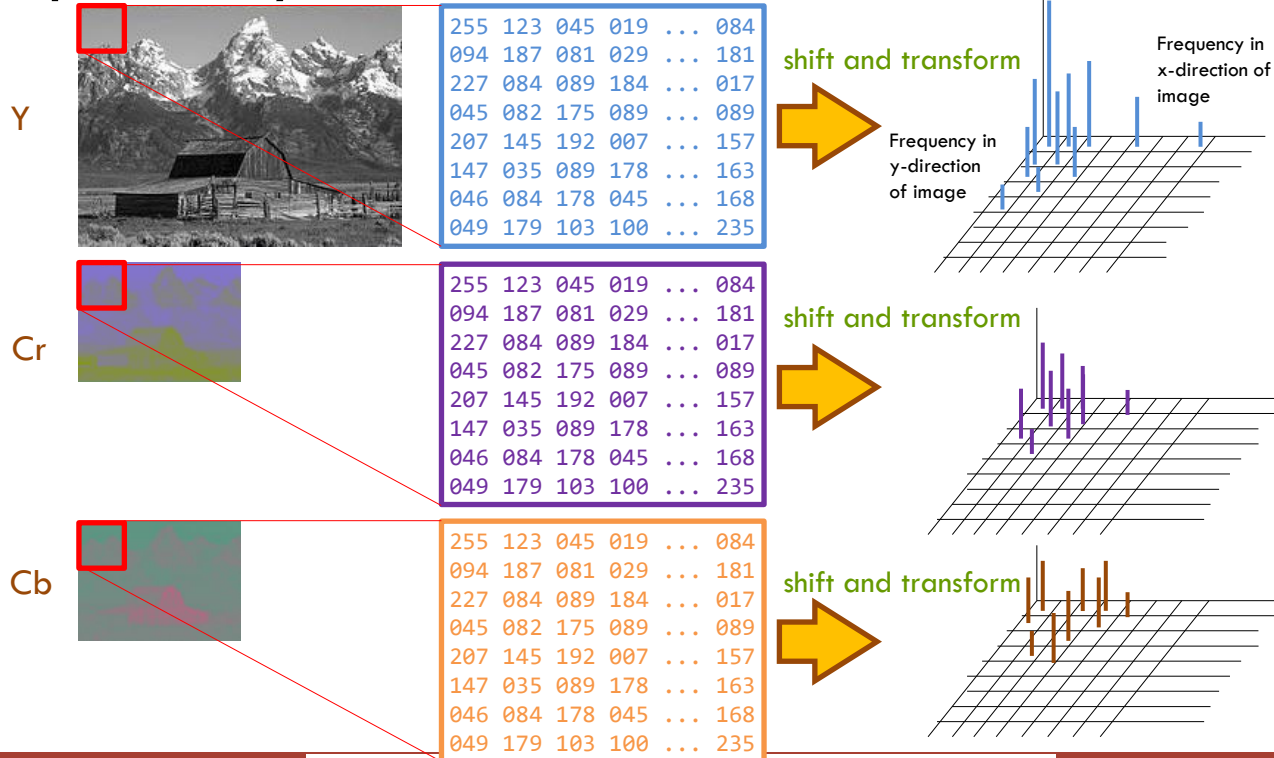
255	123	045	019	...	084
094	187	081	029	...	181
227	084	089	184	...	017
045	082	175	089	...	089
207	145	192	007	...	157
147	035	089	178	...	163
046	084	178	045	...	168
049	179	103	100	...	235

shift and transform



# JPEG Transformation

- An image is divided into  $8 \times 8$  pixel<sup>2</sup> blocks
- The Y, Cr, Cb components are transformed separately





# Understanding DCT

- For JPEG's 8x8 blocks, the values of  $I(y, x)$  are non-negative
  - DCT requires range to be centered around 0
  - A shifting of  $[0, 255]$  to  $[-128, 127]$  is required
  - It can be done by subtracting 128 for every samples in the blocks

## ■ Example:

52	55	61	66	70	61	64	73	Shifting ➔	-76	-73	-67	-62	-58	-67	-64	-55
63	59	55	90	109	85	69	72		-65	-69	-73	-38	-19	-43	-59	-56
62	59	68	113	144	104	66	73		-66	-69	-60	-15	16	-24	-62	-55
63	58	71	122	154	106	70	69		-65	-70	-57	-6	26	-22	-58	-59
67	61	68	104	126	88	68	70		-61	-67	-60	-24	-2	-40	-60	-58
79	65	60	70	77	68	58	75		-49	-63	-68	-58	-51	-60	-70	-53
85	71	64	59	55	61	65	83		-43	-57	-64	-69	-73	-67	-63	-45
87	79	69	68	65	76	78	94		-41	-49	-59	-60	-63	-52	-50	-34

## Example

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

Shifting

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

DCT

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

# What is DCT?

- The DCT converts a spatial description (amplitudes of luminance/chrominance values) of an image into frequency-based description
  - i.e. how rapid is the change of the luminance/chrominance values
- The obtained data  $F(v, u)$  are the coefficients of the 2D cosine waves
  - $F(0,0)$  is called the DC coefficient
    - The frequency is zero
    - Represents the average intensity/fundamental color of the block
  - $F(0,1)$  to  $F(7,7)$  are called the AC coefficients
    - Represent the weighing of different cosine waves
    - The larger the  $u$  and  $v$ , the higher the frequency of the cosine wave

# What do the Coefficients Imply?

- Usually an useful image has larger low frequency components than high frequency components
  - i.e.  $F(v, u)$  of small  $u$  &  $v \gg F(v, u)$  of large  $u$  &  $v$
  - $F(v, u)$  of large  $u$  and  $v$  tends to zero
  - The color data within the block is not changing at all or changing slowly
- However, if an image has larger values at high frequency components
  - The color data is changing rapidly on a short distance scale

# Quantization



- Humans are usually unable to notice the high frequency variation (rapid change on short scale) of an image
- DCT allows us to isolate those high frequency components from lower frequency components
- We can then remove some high frequency components
  - How? by quantization

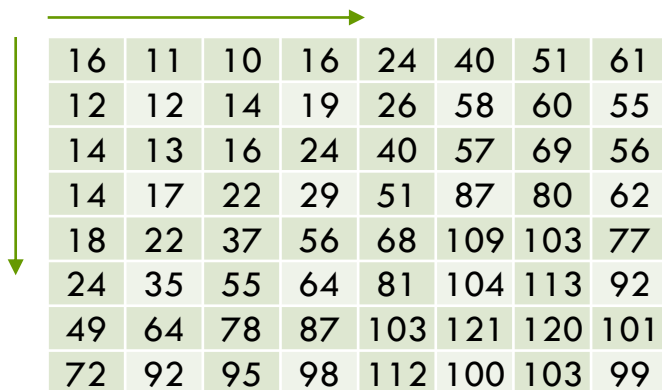
# Quantization

■ Divide  $F(v, u)$  by a matrix and round the result

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

■ For JPEG, the typical 50% quality quantization matrix  $Q$

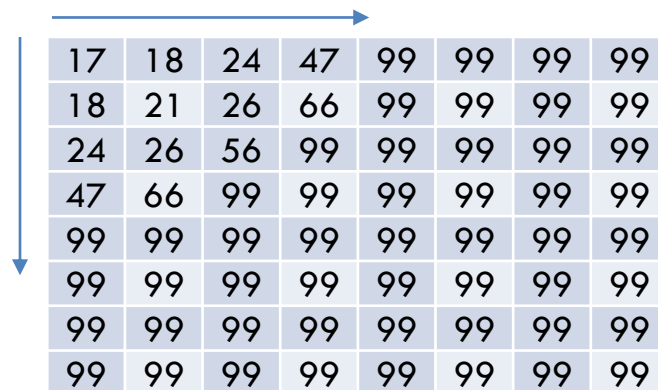
The higher the frequency in x/y direction, the larger the quantization step, the smaller the  $F_q$  values (require fewer bits to represent)



A green arrow points right above the table, and a green arrow points down to the left of the table, indicating that quantization values increase with frequency in both horizontal and vertical directions.

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

Luminance Quantization Table



A blue arrow points right above the table, and a blue arrow points down to the left of the table, indicating that quantization values increase with frequency in both horizontal and vertical directions.

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

Chrominance Quantization Table

# Quality Factor vs. Quantization

- Quality factor  $q$  is used to control the compression ratio and image quality
  - Quantization Matrix  $Q' = \frac{(100-q)}{50} Q$
  - For  $q = 50$ ,  $Q' = Q$  (i.e. default quantization)
  - For  $q = 99$ ,  $Q' = 0.02Q$
  - For  $q = 1$ ,  $Q' = 1.98Q$
- The higher the quality factor, the better the image quality (but lower compression ratio)

## Example

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

Shifting

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

DCT

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

Quantization

$$Fq(v,u) = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



# Quantization Table and Quality

- Quantization is the main cause of lossy
  - The table also controls the compression ratio
  - The larger the values in the quantization table, the greater the compression
  - But also the more the zeroes in the AC coefficients, the more the components to be removed and thus the lower the quality
  - [http://www.visengi.com/products/jpeg\\_hardware\\_encoder](http://www.visengi.com/products/jpeg_hardware_encoder)
- The human eye is insensitive to gradual changes in color but sensitive to intensity
  - So we try to ignore gradual changes in color and throw away data without the human eye noticing
  - JPEG provide two quantization tables: one for luminance, another one for chrominance blocks
  - Users may also use their own quantization table for compression, but need to store the table in the JPEG header for decoder to reconstruct the image
- After quantization, the higher AC coefficients (located on the lower right corner of Fq) tends to zero

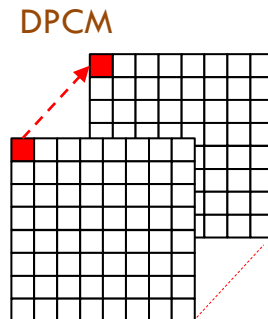
# Encoding



- After transformation and quantization, the coefficients  $Fq(v, u)$  are ready to be encoded
- However, quantized DC coefficients and quantized AC coefficients are encoded differently

# DC Coefficients

- The difference of quantized DC coefficients between successive blocks tends to be zero
  - Since the change of intensity across blocks usually relatively slowly
  - Modeling: use the quantized DC coefficient of previous block to predict the quantized DC coefficient of current block
  - Then encode the residuals of the DC coefficients
  - Called Differential Pulse Code Modulation (DPCM)



# DC Coefficients: Encoding

- Each DPCM-coded DC coefficient is represented by a pair of variable length symbols (SIZE, AMPLITUDE) where
  - SIZE is the Huffman coded no. of bits needed to represent coefficient, and
  - AMPLITUDE is the 1's complement representation of the coefficient
- Why only SIZE is Huffman coded?
  - Since AMPLITUDE can change widely, there is no much benefit in coding it
  - But SIZE does not change too much and small SIZEs occur frequently, it is worth to encode it using Huffman coding
- Finally, the Huffman table is stored in the header of the JPEG file

# AC Coefficients: Zig-Zag Scan

- Use zig-zag ordering

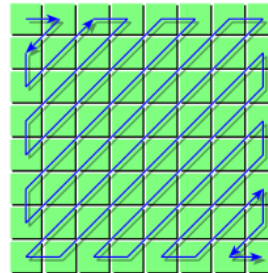
- Order the lower frequency components before the higher frequency components

- To produce maximal series of 0s at the end

- Maps 8 x 8 matrix to a 1 x 64 vector

- Example:

$$F_q(v,u) = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



$F_q(z) = \{-26, -3, 0, -3, -2, -6, 2, -4, 1, -3, 1, 1, 5, 1, 2, -1, 1, -1, 2, 0, 0, 0, 0, 0, -1, -1, 0, \dots, 0\}$

# AC Coefficients: Run-Length Encoding

- Since there are a lot of 0 AC coefficients, run-length encoding (RLE) is efficient to encode them
  - Encode a series of 0s as a (skip, value) pair, where skip is the number of zeros and value is the next non-zero component
  - Send (0, 0) as end-of-block sentinel value

■ Example:

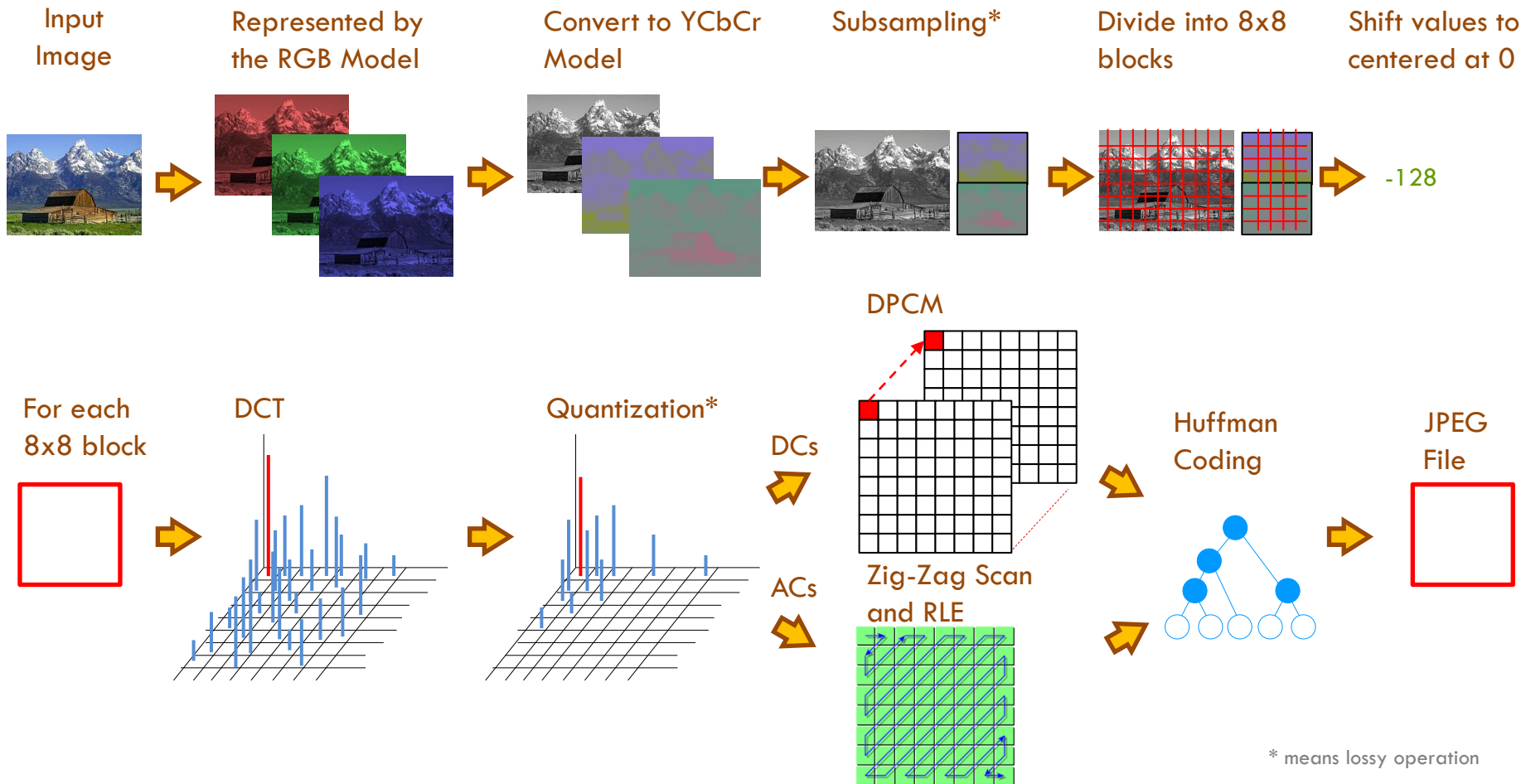
$F_q(z) = \{-26, -3, 0, -3, -2, -6, 2, -4, 1, -3, 1, 1, 5, 1, 2, -1, 1, -1, 2, 0, 0, 0, 0, 0, -1, -1, 0, \dots, 0\}$

Diagram annotations: A red arrow points from the label "DC" to the value -26. A blue arrow points from the label "(1, -3)" to the first zero (0) in the sequence. A green arrow points from the label "(5, -1)" to the fifth zero (0) in the sequence.

$RLE = \{(0, -3), (1, -3), (0, -2), (0, -6), (0, 2), (0, -4), (0, 1), (0, -3), (0, 1), (0, 1), (0, 5), (0, 1), (0, 2), (0, -1), (0, 1), (0, -1), (0, 2), (5, -1), (0, -1), (0, 0)\}$

- Finally, apply Huffman coding on the RLE of AC coefficients

# JPEG: Summary



# JPEG Modes

- There are four modes defined in the JPEG standards
  - Lossless Mode
  - Baseline Mode
  - Progressive Mode
  - Hierarchical Mode
- Progressive Mode
  - Allows a coarse version of an image to be transmitted at a low rate, which is then progressively improved over subsequent transmissions
  - Good for large images that will be displayed while downloading over a slow connection, allowing a reasonable preview after receiving only a portion of the data



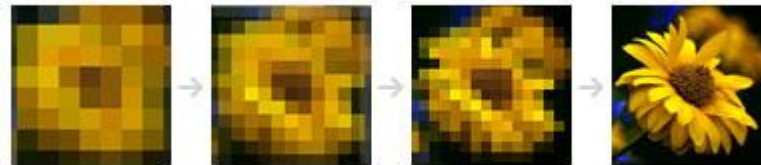
# Progressive Mode

- In contrast, the standard baseline mode provides the blocks one by one, from left to right and top to bottom

Baseline  
Mode



Progressive  
Mode



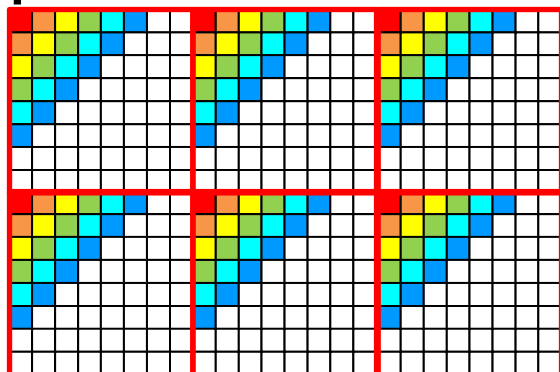
# Progressive Mode



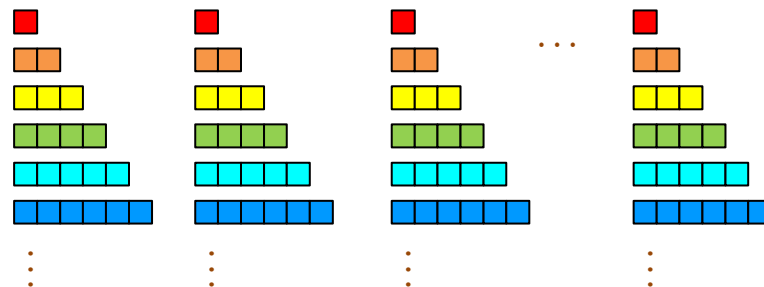
- There are two implementations of the progressive mode
  - Spectral Selection: for each block, send DC and first few AC coefficients first, then gradually some more ACs
  - Successive Approximation: All coefficients are sent few most significant bits in the first scan, then the next few bits of all coefficients in the second scan and so on

# Progressive Mode

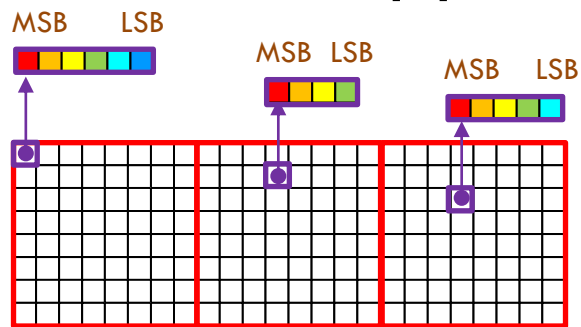
## Spectral Selection



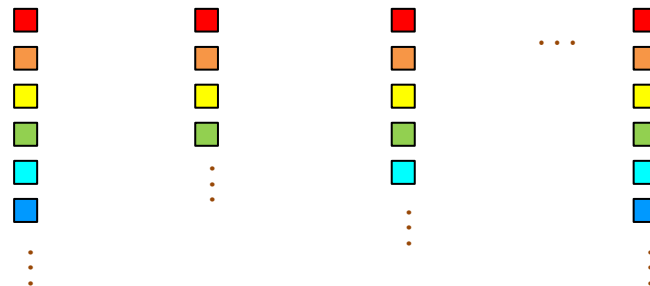
First Scan  
Second Scan  
Third Scan  
...



## Successive Approximation



First Scan  
Second Scan  
Third Scan  
...



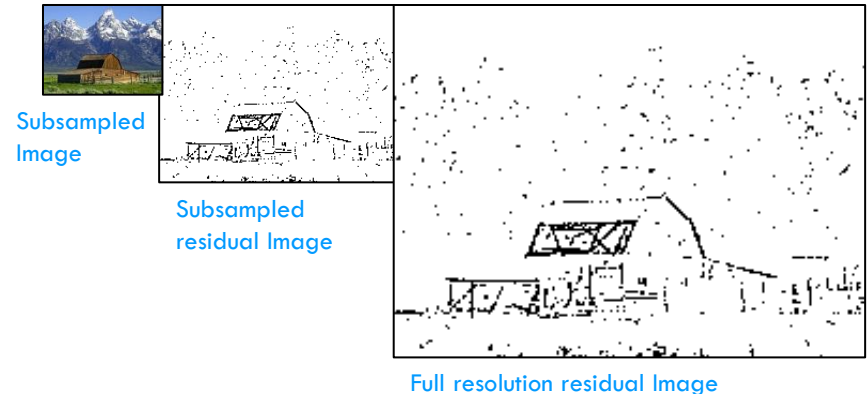
# Hierarchical Mode

- The hierarchical mode encodes an image in multiple resolutions
- The decoder at the receiving end can choose the optimum resolution depending on the target's capabilities
- This is particularly useful for applications where multiple resolutions are needed at different times

Source Image



Hierarchical Mode JPEG



# Hierarchical Mode

- In hierarchical mode, the image is first subsampled by 2
  - The new size is reduced by 2 horizontally and vertically
  - The reduced image is encoded
- The encoded reduced-size image is decoded and upsampled by 2 horizontally and vertically
  - This upsampled image is used as a prediction of the original image at this resolution
  - The residual image is encoded
- This encode and decode processes are repeated until the original image at full resolution has been encoded
- Since the higher resolution images are coded as differences from the next smaller image, they generally require fewer bits than they would if encoded independently