# Resources:

- Dr Rawat (UCF)
- Ms. Adeela (PhD scholar PUCIT)
- Dr Shree Nayar (Columbia University )

# Lecture 3

# Outline

- Image digitization
    - Sampling
    - Quantization
- Image as a function
- Image processing
    - Histogram
        - Histogram equalization

# Need of digitization

- Every scene around us is a continuous image that can be represented by infinite number of image points (resolution)
- Each such image points may contain infinitely many possible intensities
- That needing an infinite number of bits for storage
- Theory of real number: Between any 2 given points there infinite number of points
- Obviously such a representation is not possible in any digital computer

# Digitization

Issues with continuous image

- Infinite sizes values between two real numbers
- Infinite intensity values between two real numbers

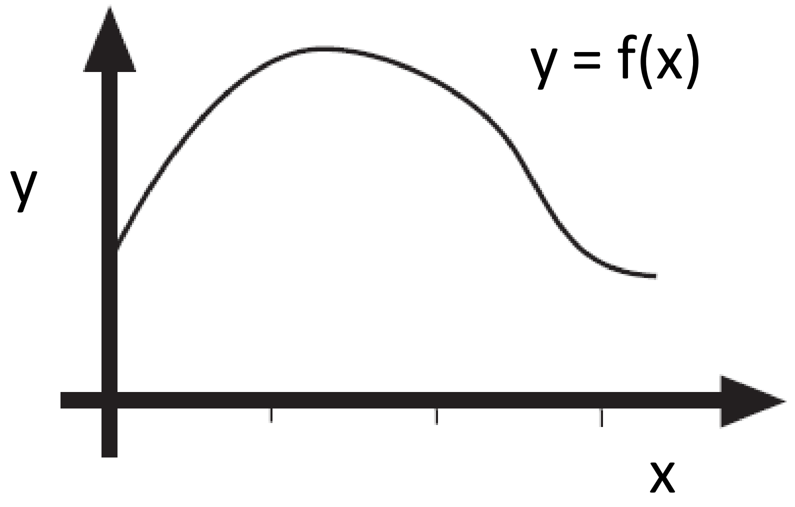
- The process of transforming *continuous space* into *discrete space* is called digitization
- Vision algorithms use a discrete form of the images

# Digitization

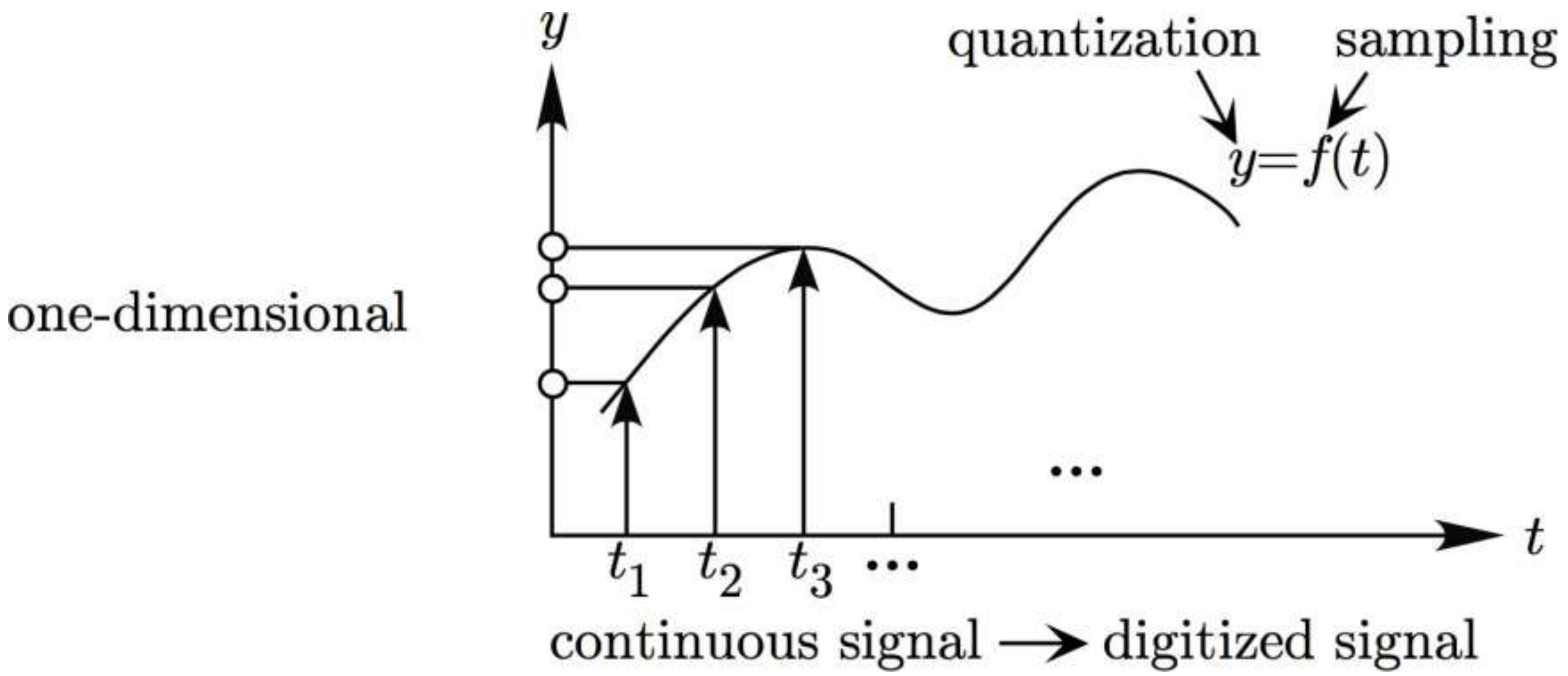
- Function

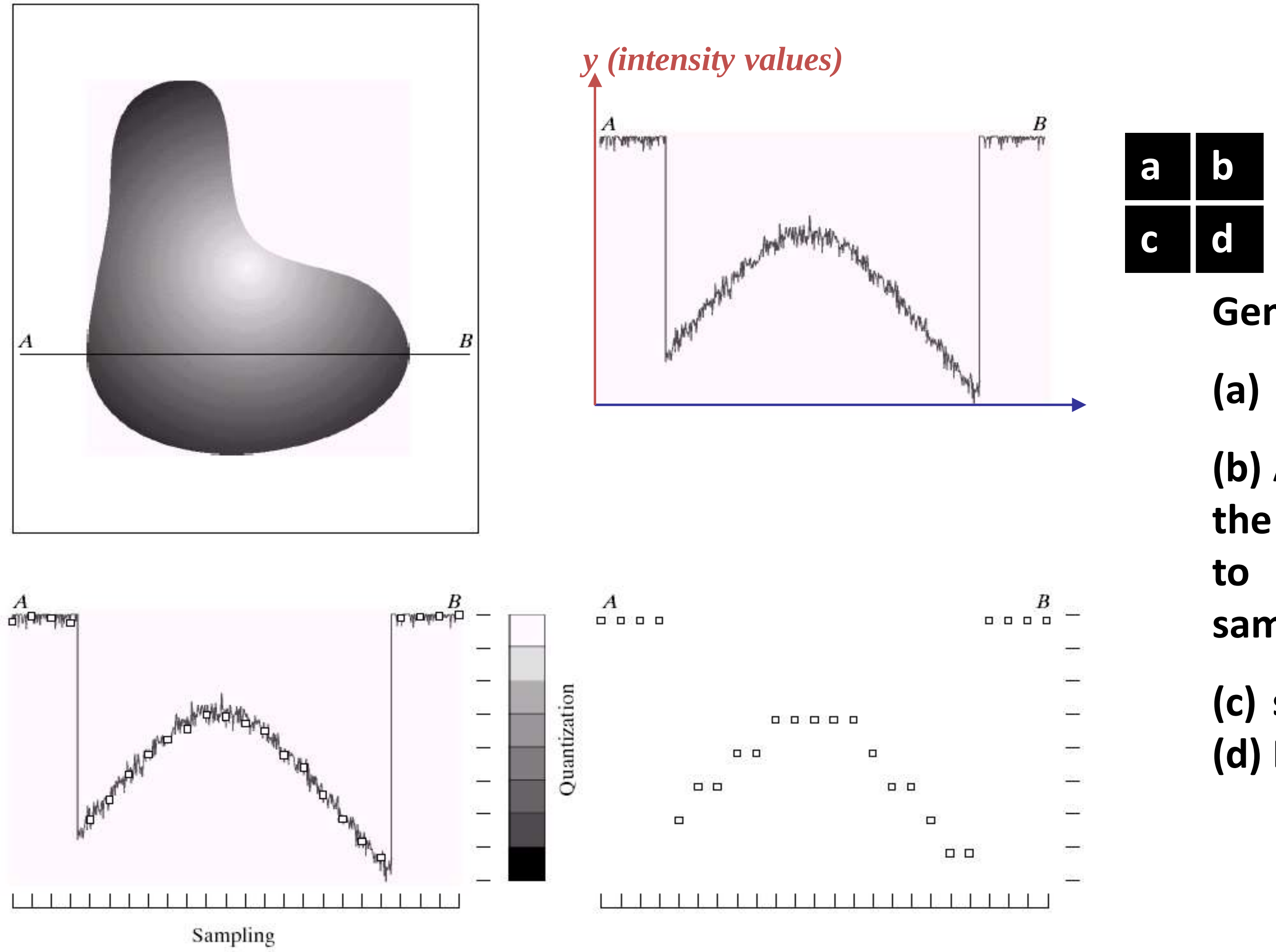
  y = f(x)

- Domain of a function(x)
- Range of a function (y=f(x))
- Sampling
  - Discretization of domain
- Quantization
  - Discretization of range



# Digitization of 1D function






**Generating a digital image.**

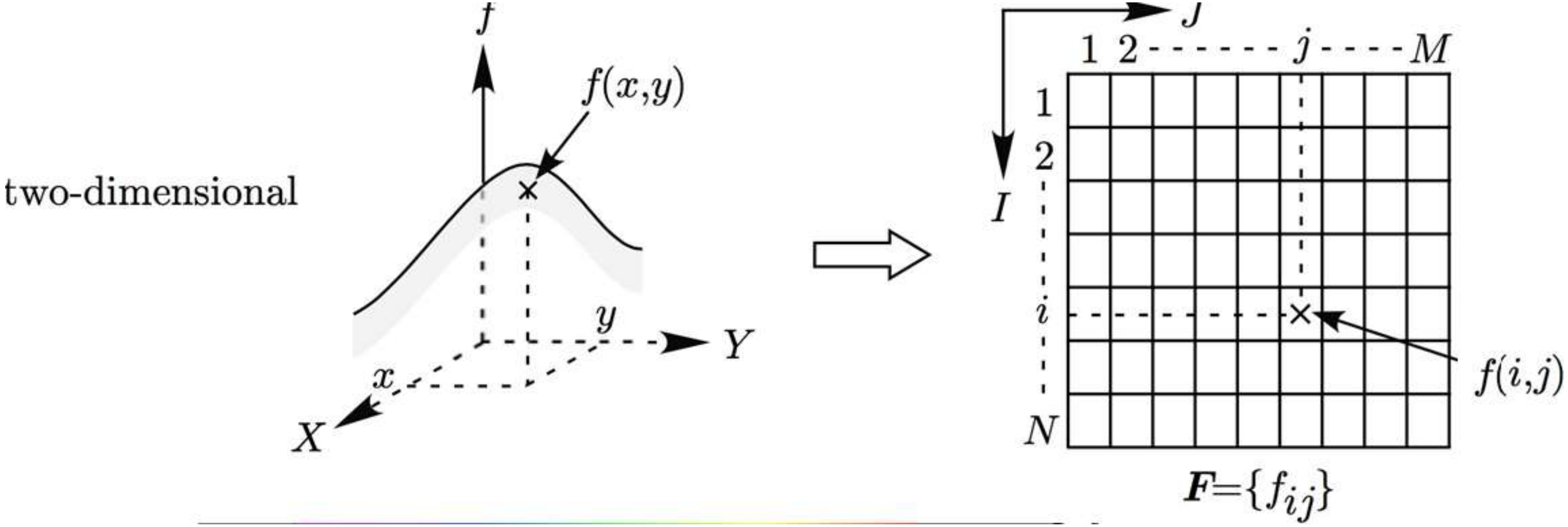
**(a) Continuous image.**

**(b) A scaling line from A to B in the continuous image is used to illustrate the concepts of sampling and quantization.**

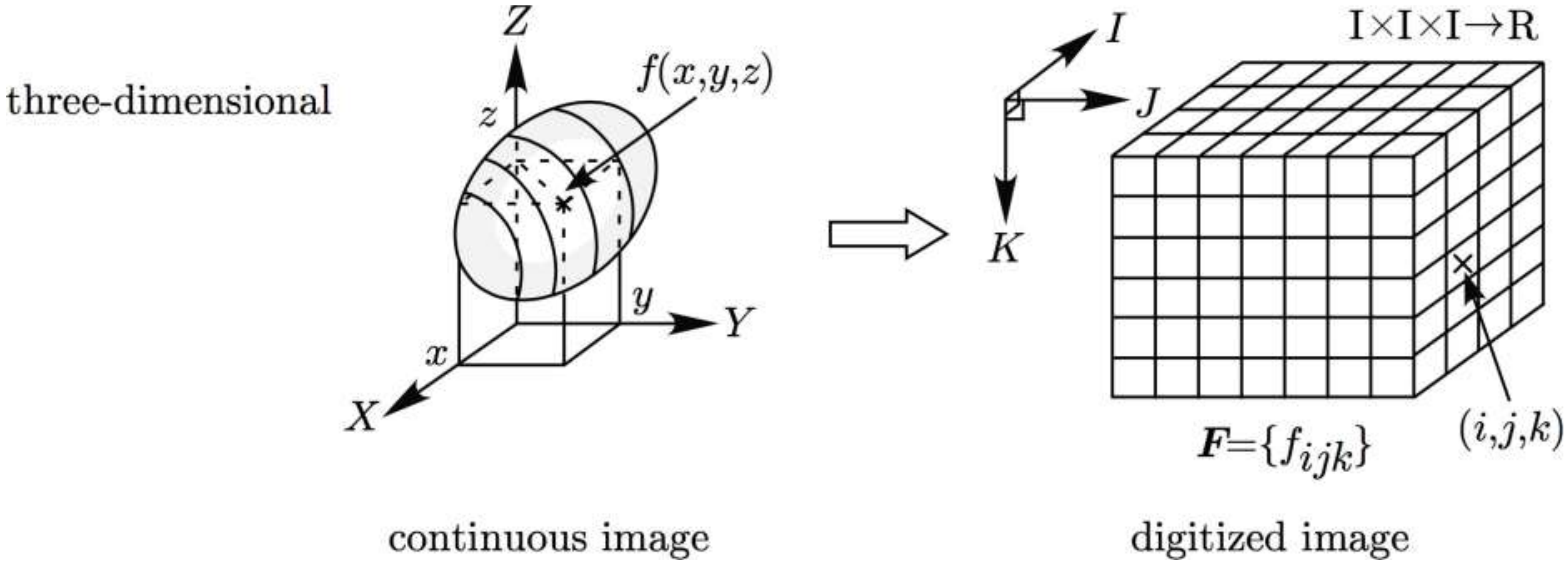
**(c) sampling and quantization.**
**(d) Digital scan line.**

Credit : https://sisu.ut.ee/imageprocessing/book/2

# Digitization of 2D function

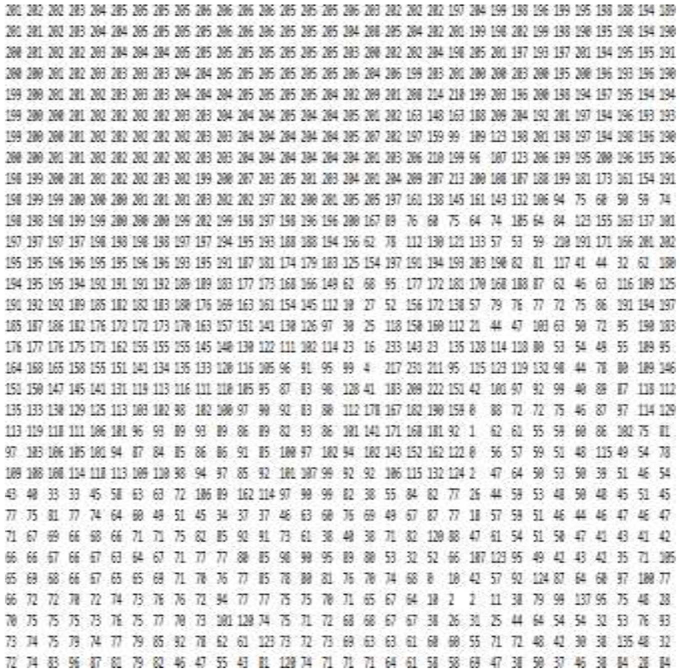


# Digitization of 3D function



# Image as 2D array

(a) Image

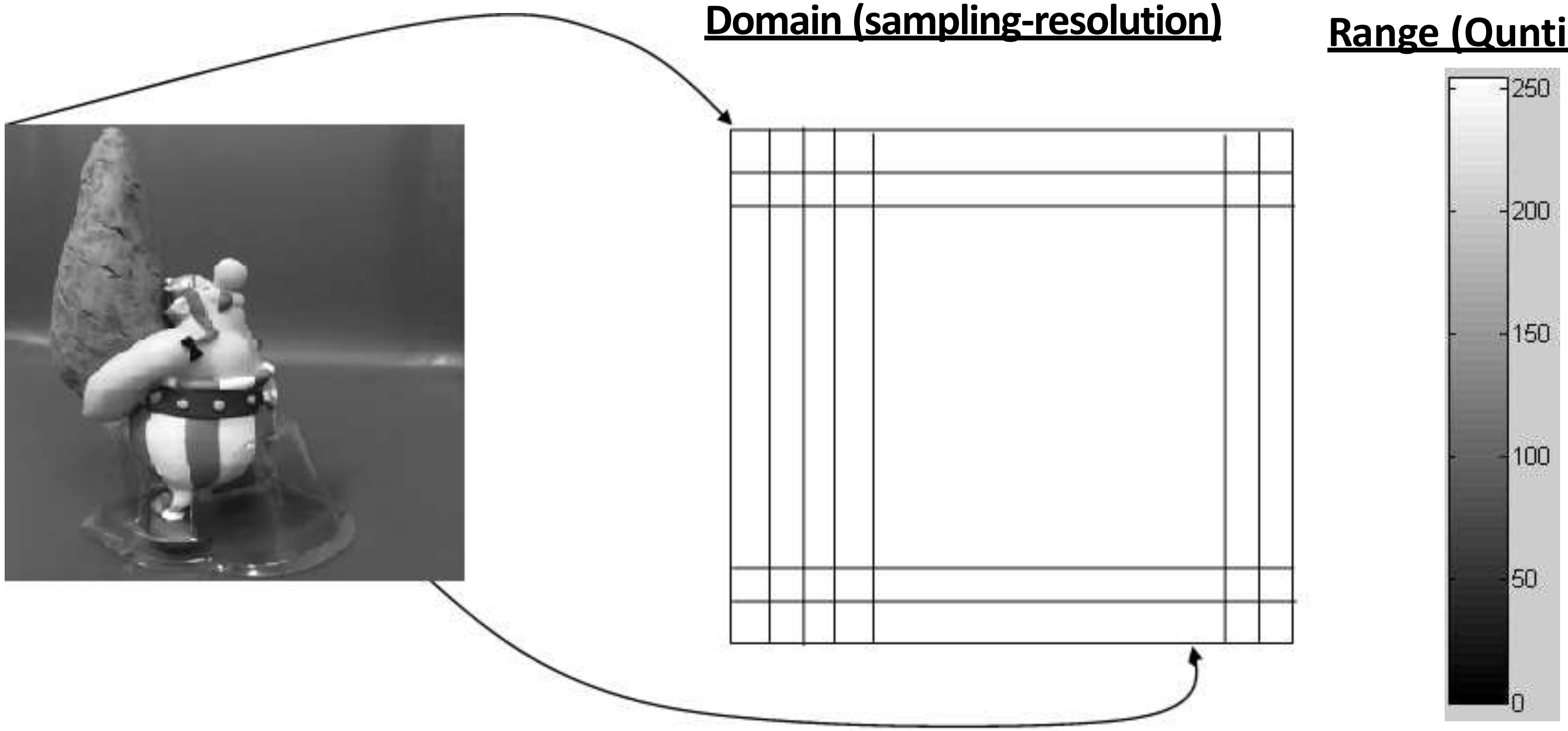


(b) as 2D array

# Digitization of image in 2D array

**Domain (sampling-resolution)**

**Range (Quntization- Intensity value)**

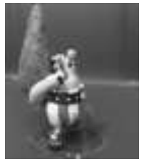


# Sampling

1024

512

256



128

64

32

# Resize the sampling images

1024

512

256

128

64

32

**Python: `cv2.resize(img, (1024, 1024))`**

# Quantization


8-bit


7-bit


6-bit


5-bit


4-bit


3-bit


2-bit


1-bit

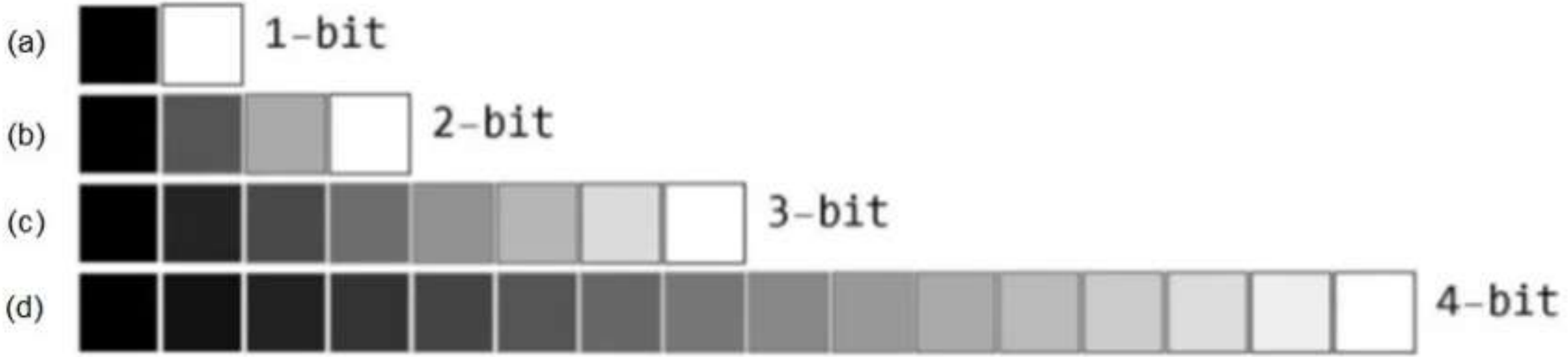
cv2.kmeans()

k= 2 for creating 2 cololr image using the **1-bit level image.**

K= 4 for creating 4 color image using **2-bits level image**

# Quantization -Image Intensity

The number of bits used to store that value of the pixel identifies the number of grey levels (or possible grey level values) used to describe a pixel.

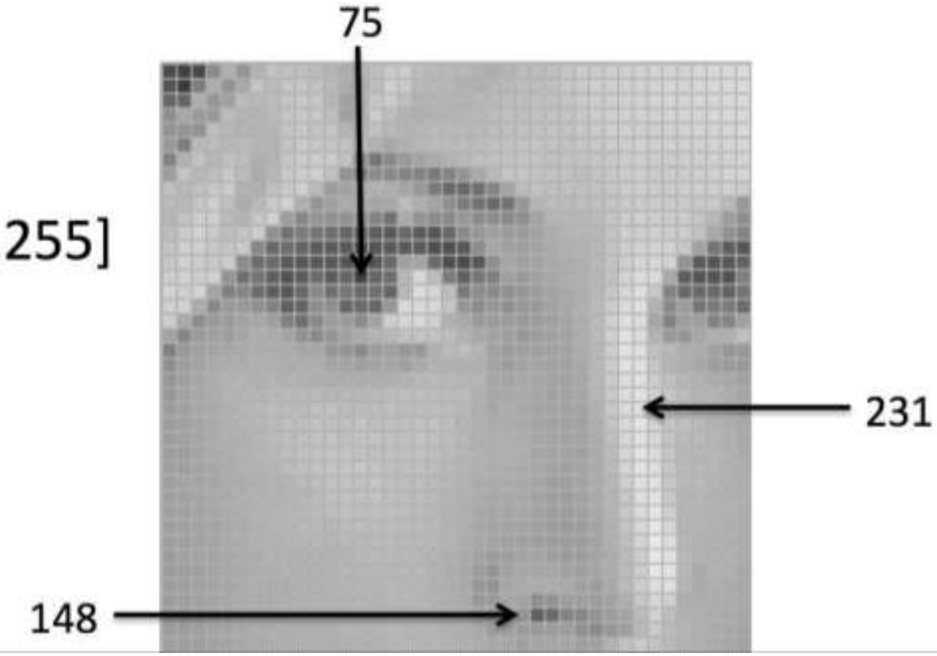



Available pixel intensities for, 2-bit, 3-bit, and 4-bit image data
1-bit : 2^1 =2 grey lever
2-bit : 2^2 =4
3-bit : 2^3 =8
4-bit : 2^4 =16
Generally gray images uses 8-bit : 2^6 =256

# Intensity values after quantization of grayscale image

– Pixel value:

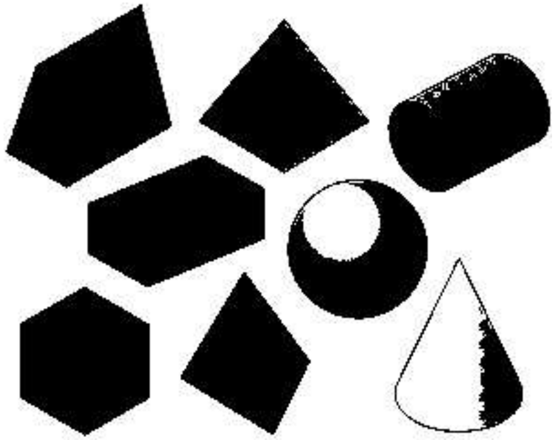
- “grayscale” (or “intensity”): [0,255]



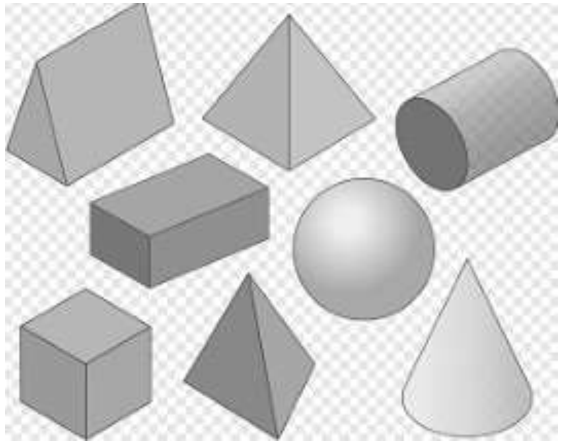
# Types of Images

- Binary
- Gray-scale
- Color

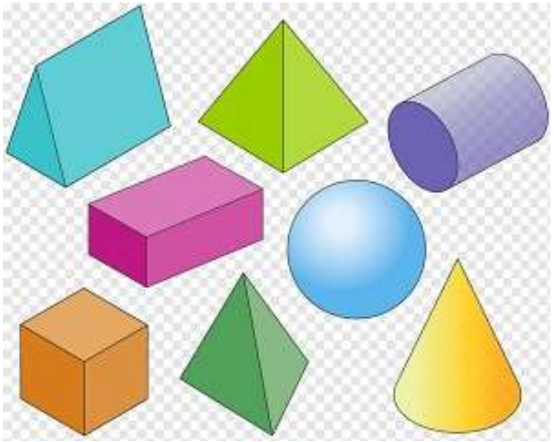
# Types of Images



Binary



Gray-scale



Color

# Binary Images

- A binary image is a digital image that has only two possible values for each pixel
- Binary images are also called *bi-level* or *two-level (1bit)*
- Binary images often arise as a result of certain operations such as segmentation, thresholding

# Binary Image Representation

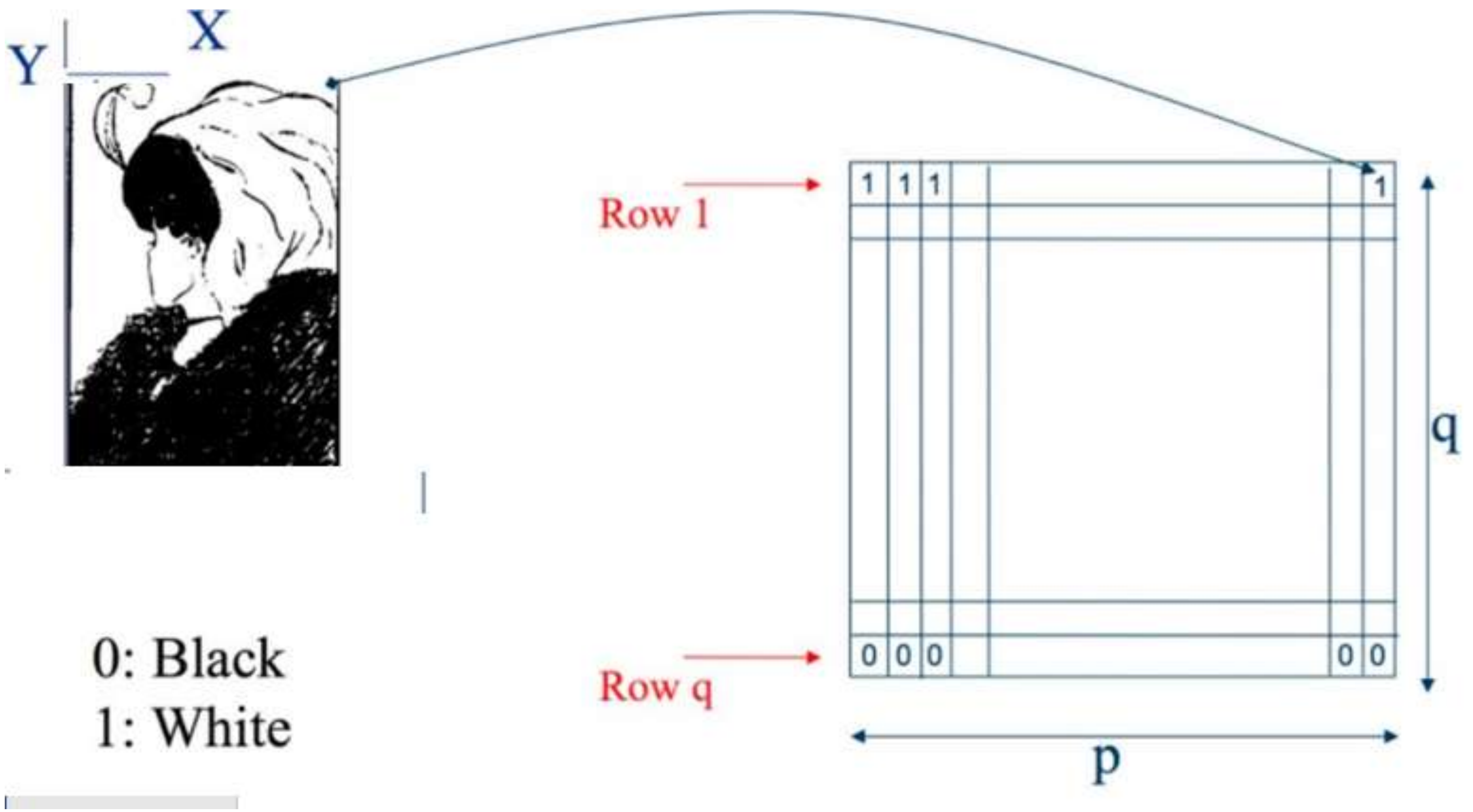



Image credits: Ulas Bagci

# Gray-scale Images

- A gray-scale image is composed of shades of gray
- Shades of gray vary from black as the weakest intensity to the white as strongest
- The values of image intensity range from 0 to 255 (8 bits)

# Gray-scale Image Representation

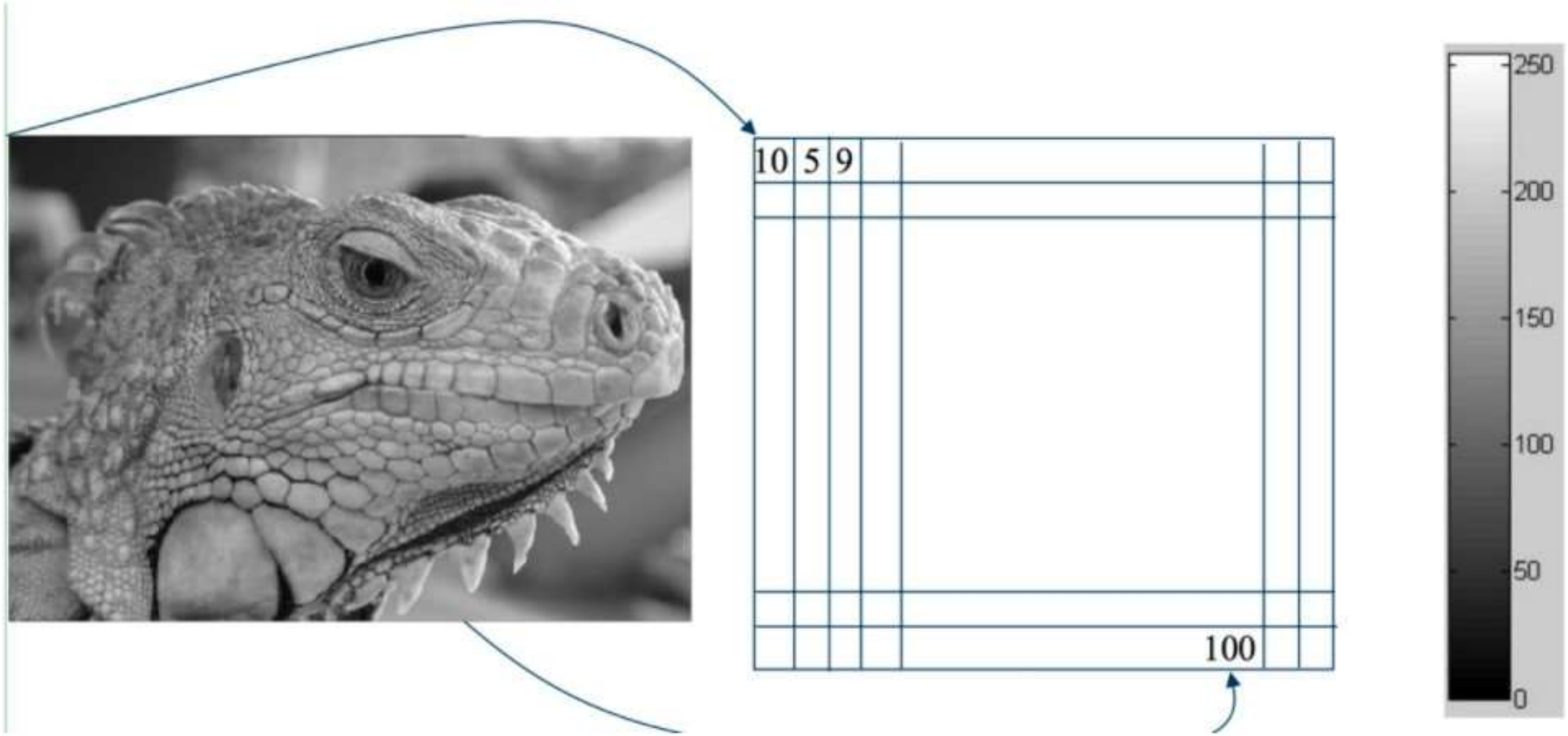



Image credits: Ulas Bagci

# Color Images

- Color image is stored as row × cols × 3 array (3 2D  array) that defines red,  green, and blue color components for each individual pixel

- This model is based on a Cartesian coordinate system

# Color Image Representation

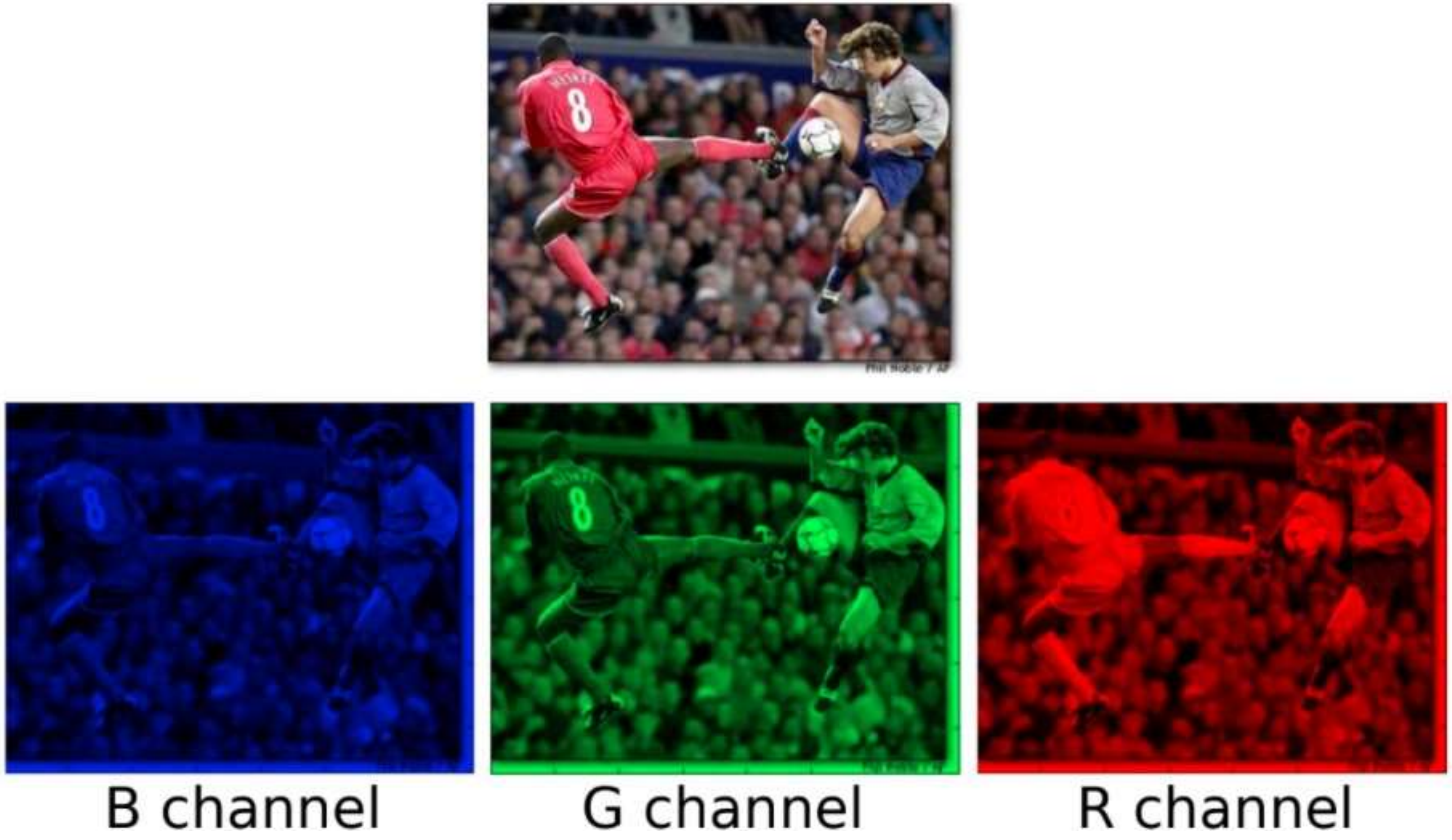
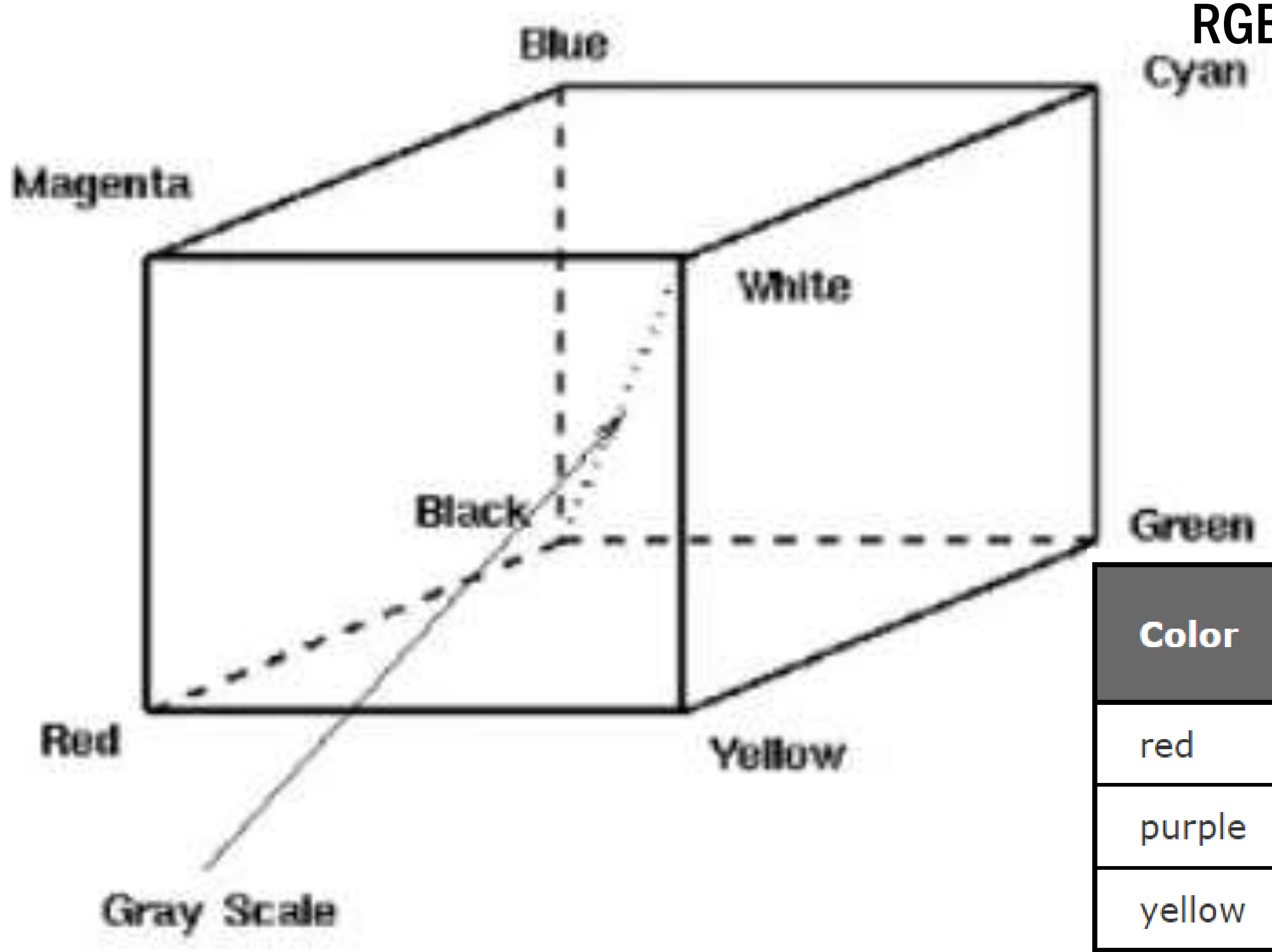



Image credits: Ulas Bagci

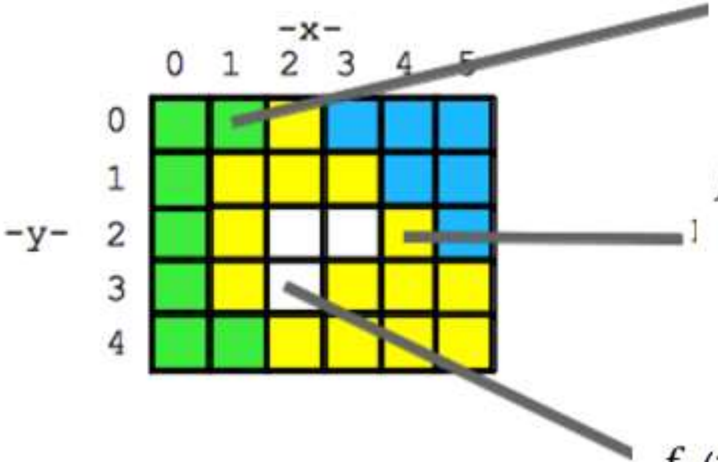
## RGB color cube: mixture of RGB color intensities




| Color | Red number | Green number | Blue numbe |
|---|---|---|---|
| red | 255 | 0 | 0 |
| purple | 255 | 0 | 255 |
| yellow | 255 | 255 | 0 |
| dark yellow | 100 | 100 | 0 |
| white | 255 | 255 | 255 |
| black | 0 | 0 | 0 |

# Color image

## Pixel values




$$f(0,1) = \begin{bmatrix} 6 \\ 250 \\ 7 \end{bmatrix}$$

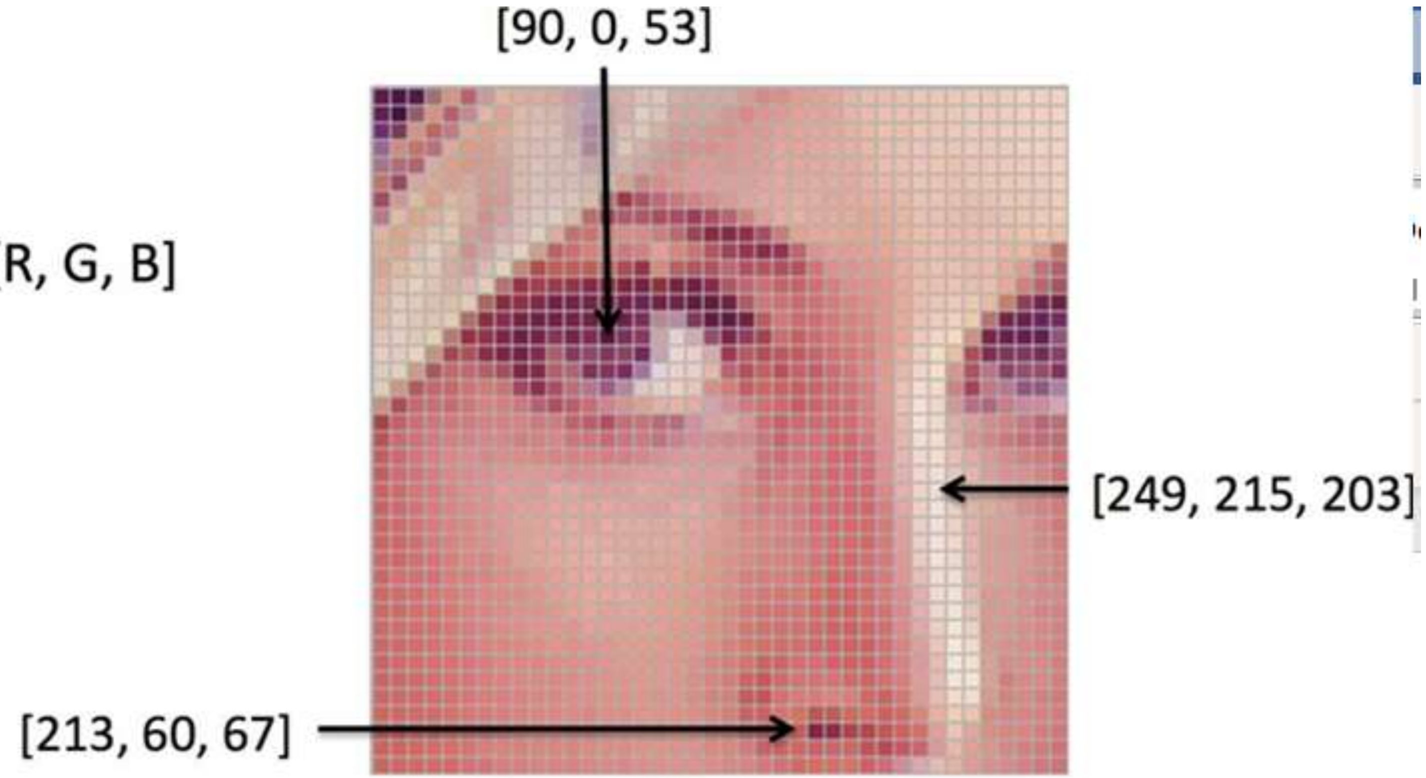
(i.e. shade of green)

$$f(2,4) = \begin{bmatrix} 241 \\ 252 \\ 23 \end{bmatrix}$$

$$f(3,2) = \begin{bmatrix} 247 \\ 250 \\ 237 \end{bmatrix}$$ red:247 green:250 blue:237

# Color image

– Pixel value:
  - "color"
    – RGB: [R, G, B]



Summary of color intensities

# Summary: Color intensities for Gray Scalar and Binary images

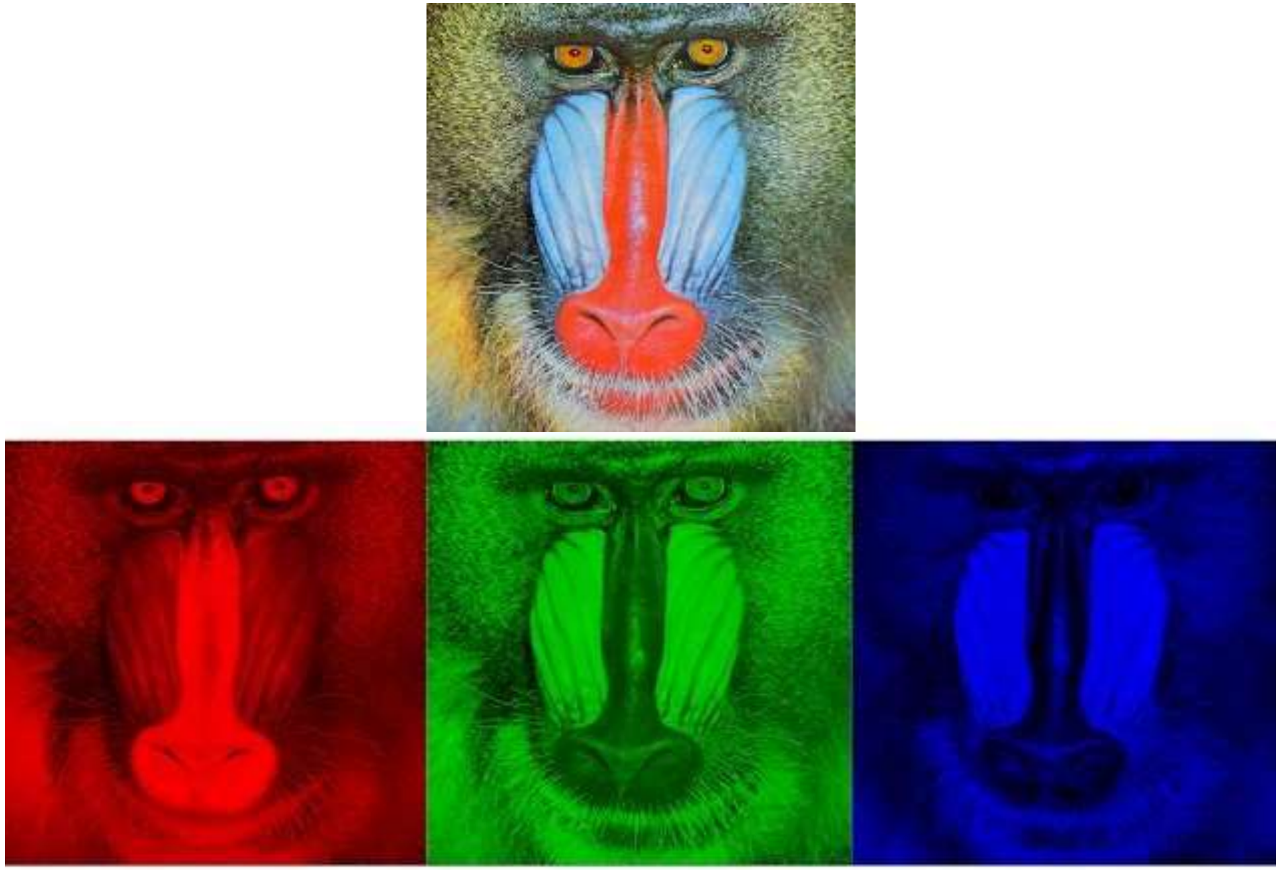
- A scalar image has integer values

$$u \in \{0, 1, ..., 2^a - 1\}$$

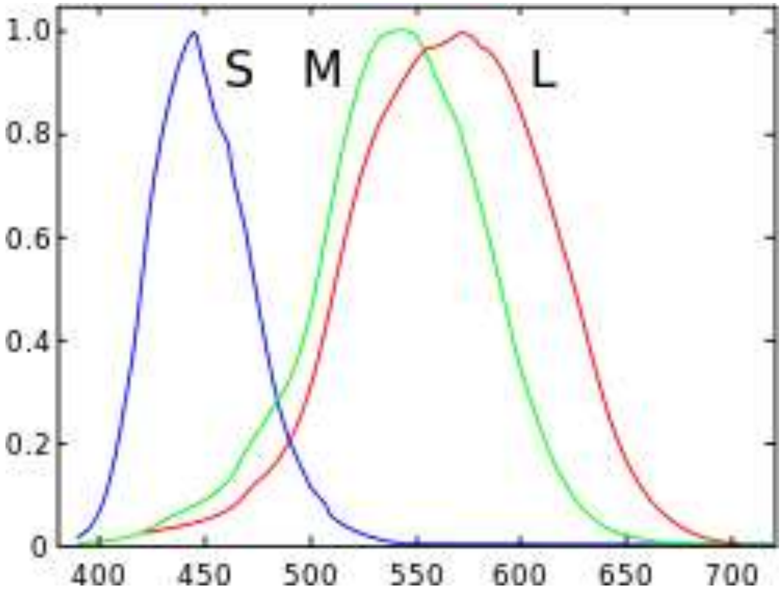
- a: level (bit)
- Ex. If 8 bit (a=8)
  - image spans from 0 to 255
  - 0 black and 255 white
- Ex. If 1 bit (a=1)
  - it is binary image
  - 0 and 1 only

# Image Type: RGB (red, green, blue)

- Each channel spans a-bit values.



Human Cone-cells (normalized) responsivity spectra



# What is "Color"?

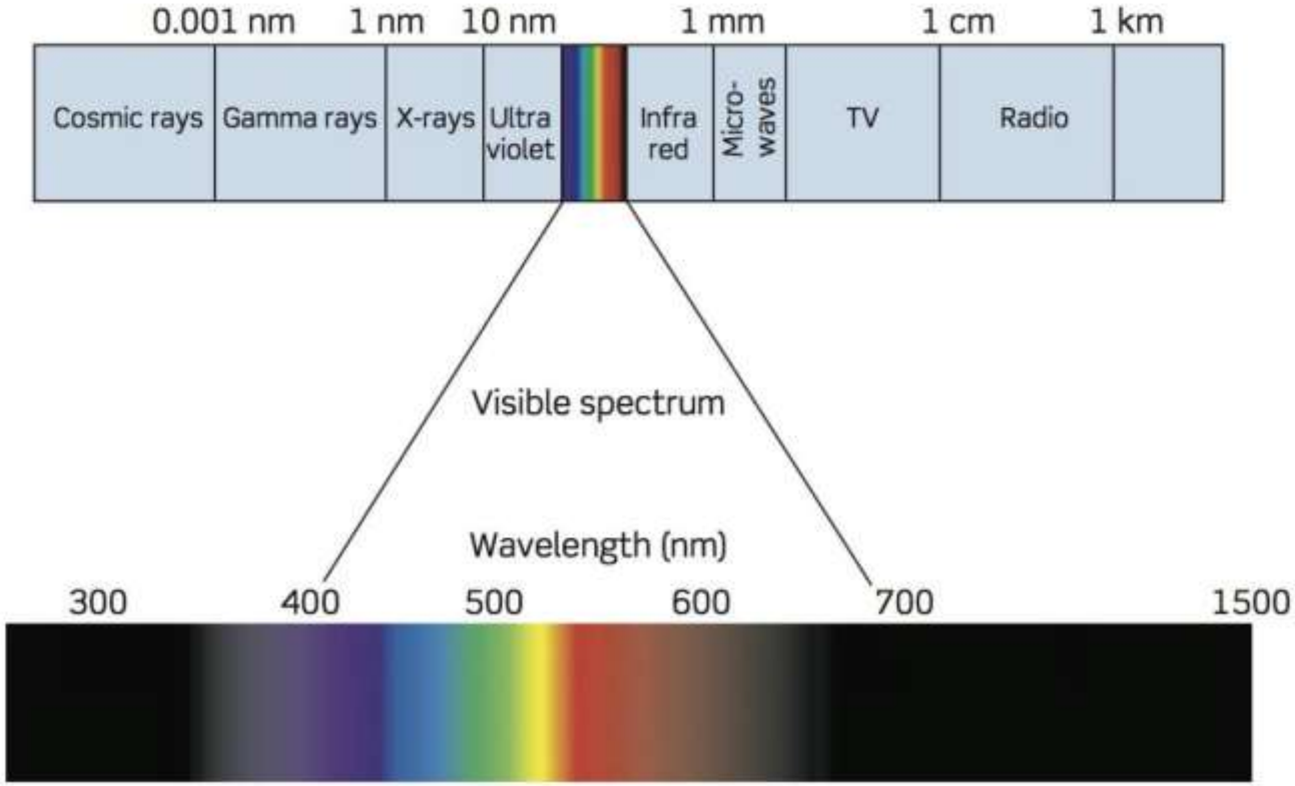
Human Response to different wavelengths

Visible light:

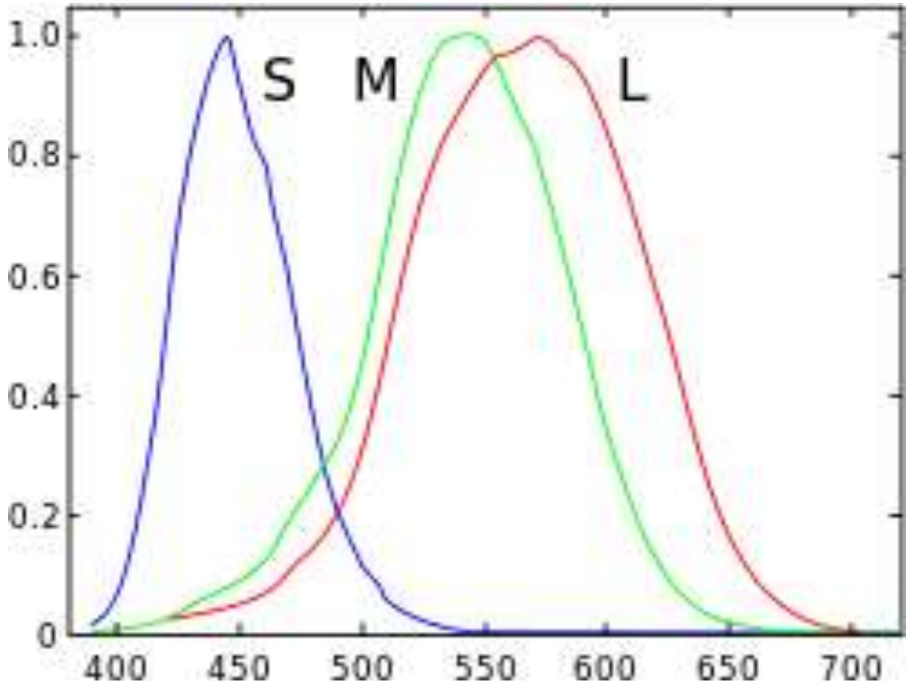

Do We recover spectral distribution $p(\lambda)$?

Sensors in the human eye: Rods & Cones

# Color



0.001 nm
1 nm
10 nm
1 mm
1 cm
1 km
Cosmic rays
Gamma rays
X-rays
Ultra violet
Infra red
Micro-waves
TV
Radio
Visible spectrum
Wavelength (nm)
300
400
500
600
700
1500

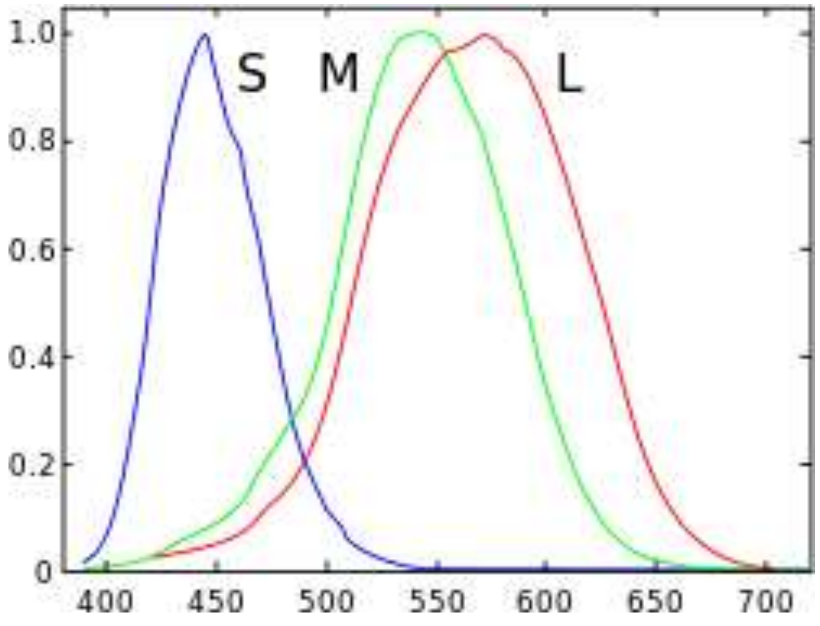



S
M
L
1.0
0.8
0.6
0.4
0.2
0
400
450
500
550
600
650
700

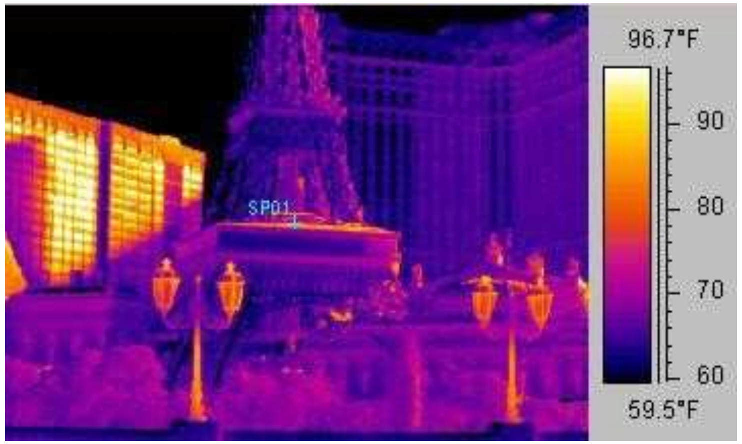
# Color

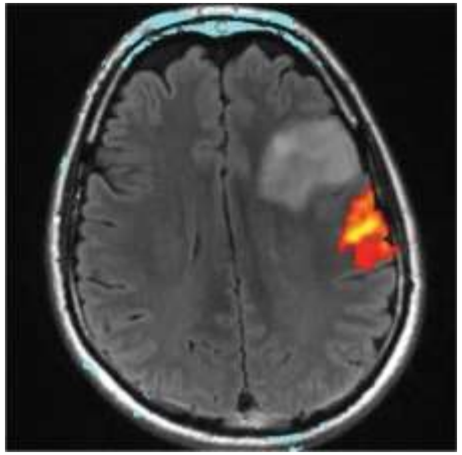
- If there is no light, there is no color!
- Human vision can only discriminate a few dozens of grey levels on a screen, but hundreds of thousands of different colors.

- RED -> ~625 to 780 nm [long wavelength]
- ORANGE -> ~ 590 to 625 nm [long wavelength]
- YELLOW -> ~565 to 590 nm [middle range wavelength]
- GREEN -> ~ 500 to 565 nm [middle range wavelength]
- CYAN -> ~485 to 500 nm [ middle range wavelength]
- BLUE -> ~440 to 485 nm [short wavelength]
- VIOLET -> ~330 to 440 nm [very short wavelength]



# Image (2D matrix) – other examples



Danny Alexander

# Image as Function

- $f(x, y)$ gives the intensity at position $(x, y)$
  - Single intensity information
- A color image is just three functions pasted together
  - $f(x, y) = [f_r(x, y), f_g(x, y), f_b(x, y)]$
    - Vector information

# Image processing

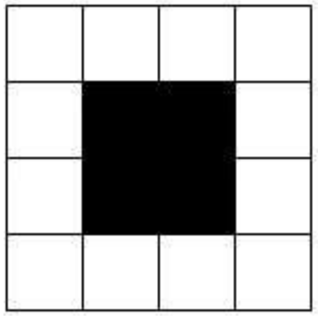
- Image processing usually defines a new image in terms of the existing Image **f**
- Image processing involves converting an original image into a modified image that is **easier to analyze**.
- Image processing improves images by **fixing problems like noise, motion blur, and defocus blur**.
- **$f'(x, y) = g(f(x, y))$** is an example of a simple operation that transforms each image pixel individually

# Histograms

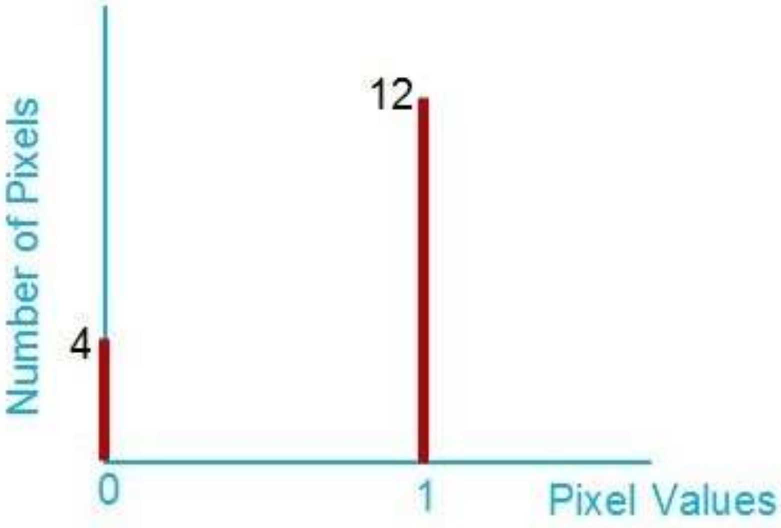
- An image histogram is a **graphical representation of pixel intensities count**
- The x-axis shows the potential intensity values, while the y-axis displays their corresponding counts.

# Histogram of a Binary Image

- The first line in the histogram, at gray level 0, shows 4 black pixels in the image.
- The next line indicates 12 white pixels.



(a)

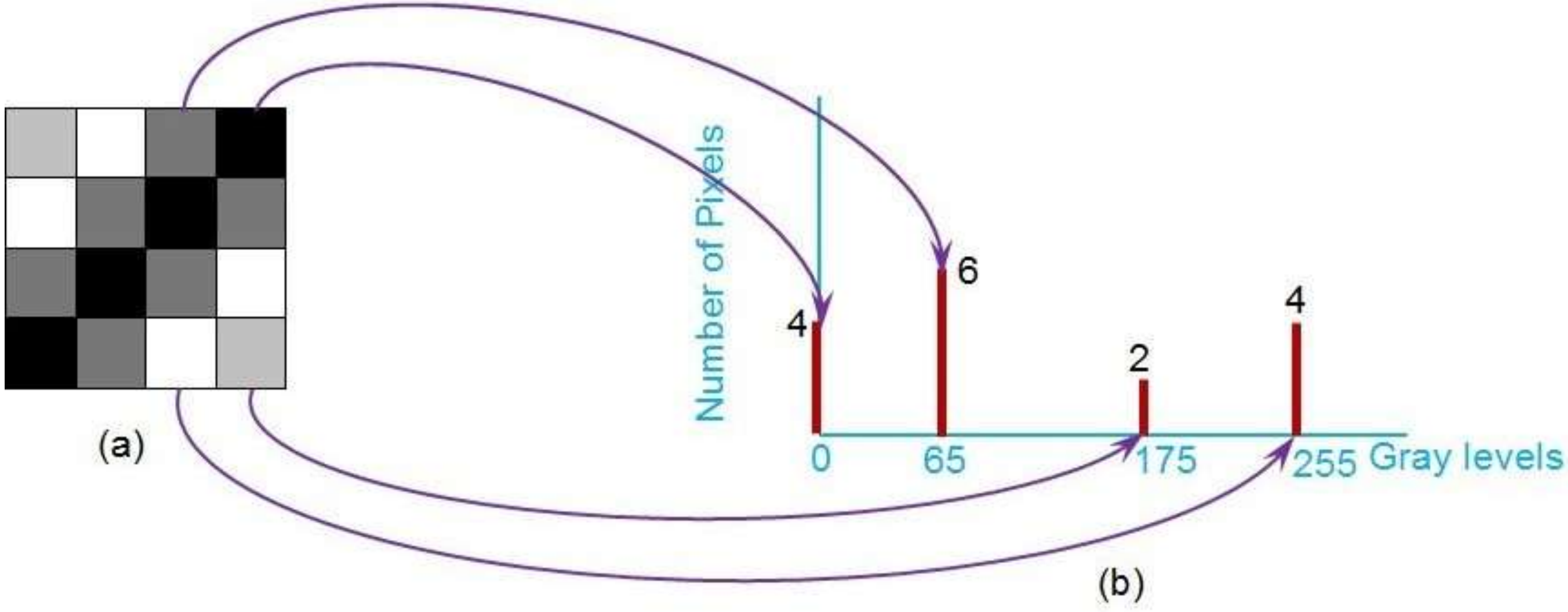



(b)

Figure credits: Sneha H.L.

# Histogram of a Gray-scale Image

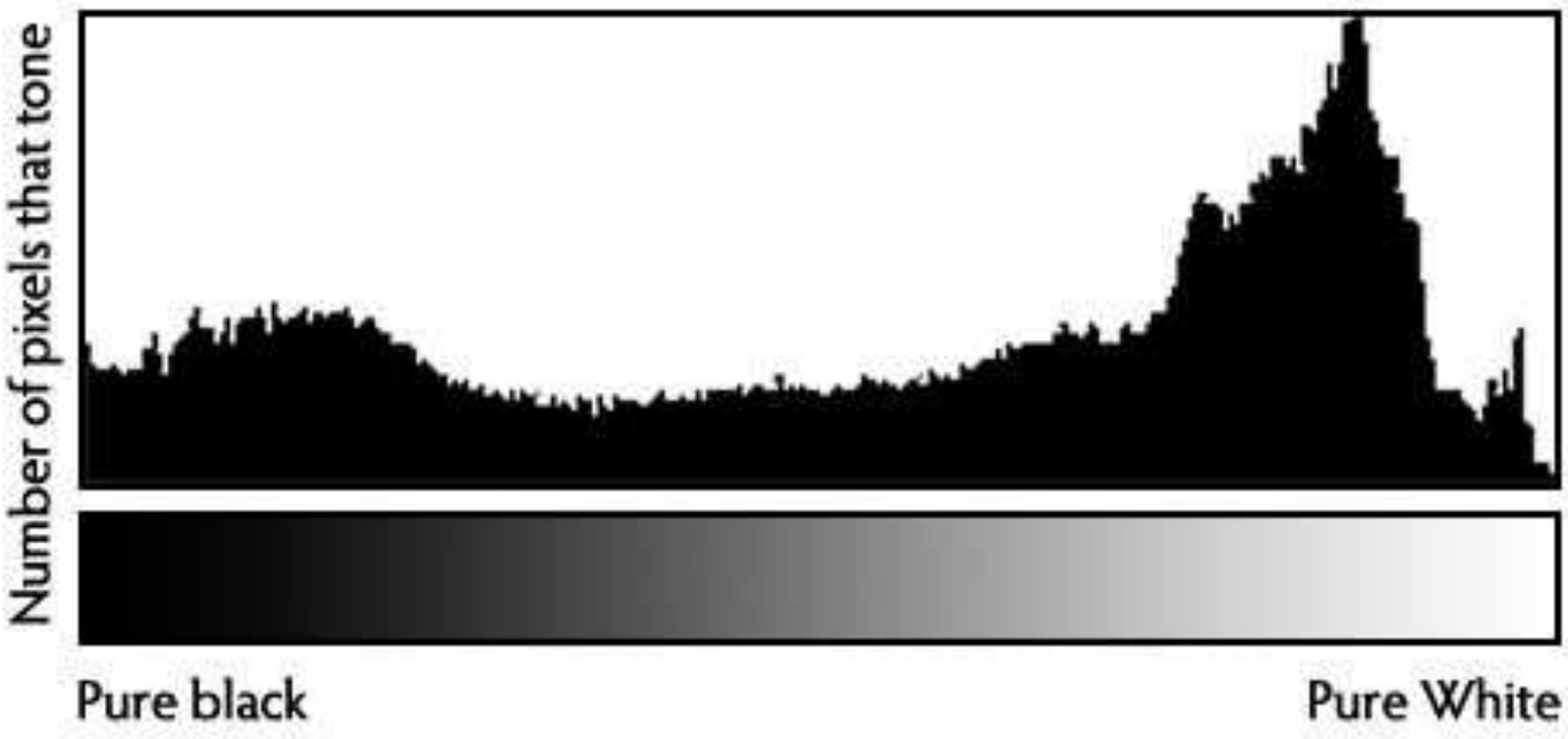
- The four-pixel intensities of this image are represented by the four vertical lines of the associated histogram. Pure black, Pure white, dark gray, light gray




Here the x-axis values span from 0 to 255, which means that there are 256 (= 2^8) possible pixel intensities.

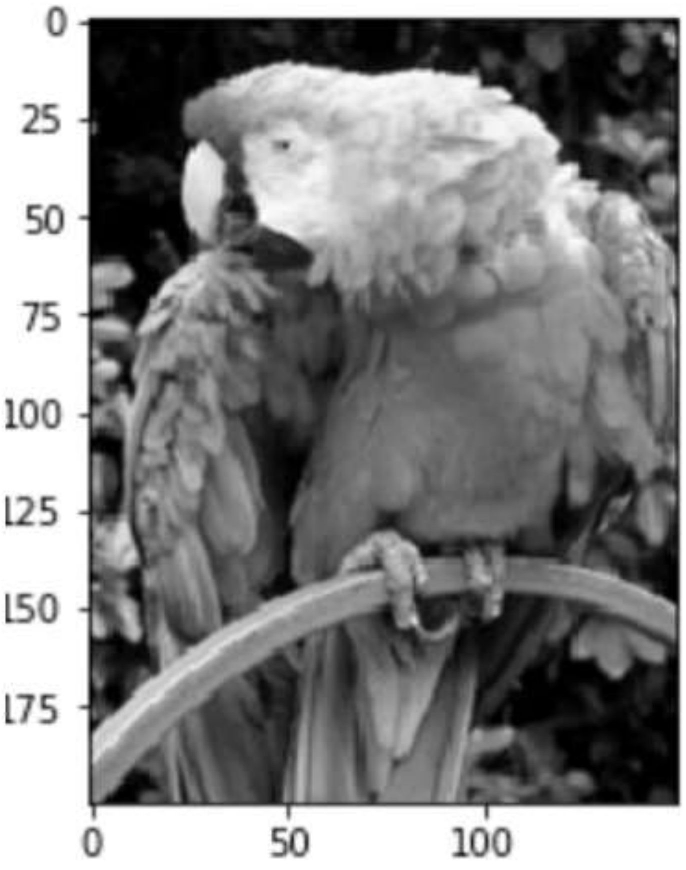
Figure credits: Sneha H.L.

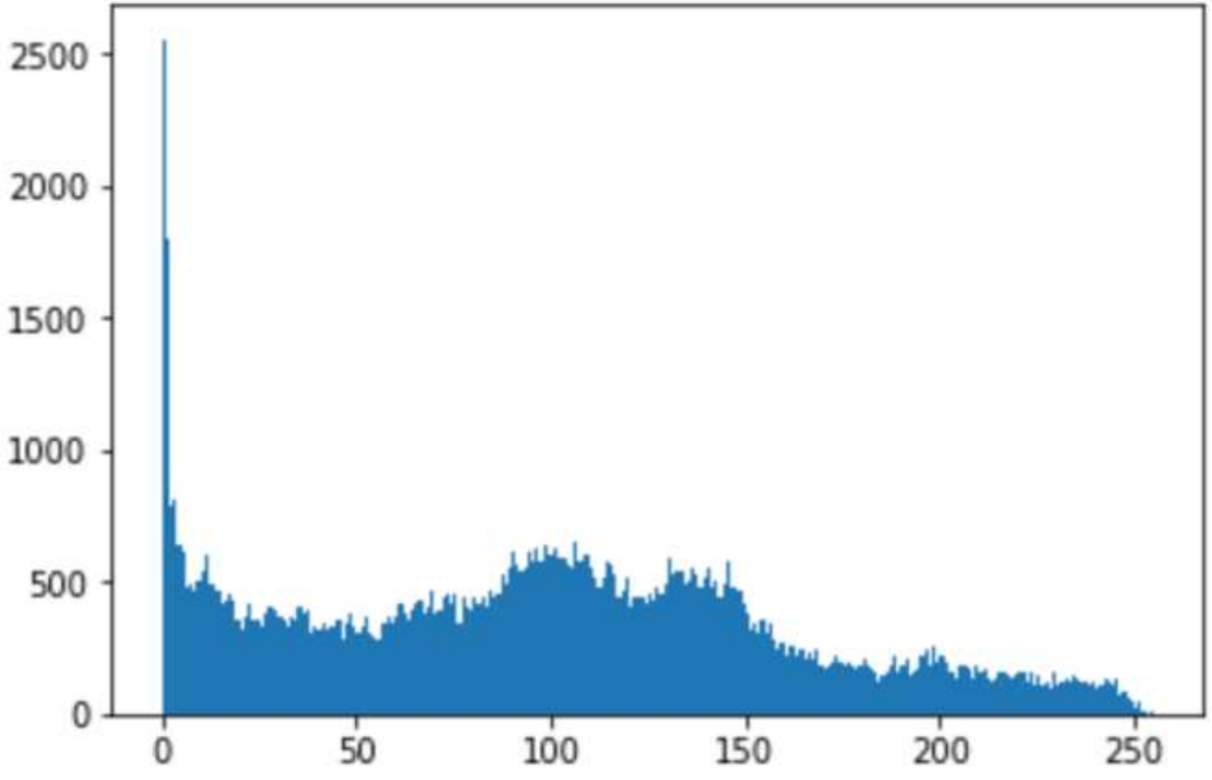
# Image Histogram- for 8 bit image



# Python code to plot histogram

```
import   cv2 as cv
import   matplotlib.pyplot as plt
img = cv.imread('parrot.png')
plt.imshow(img)
plt.hist(img.ravel(), bins =255)
plt.show()
```



# Histogram Example

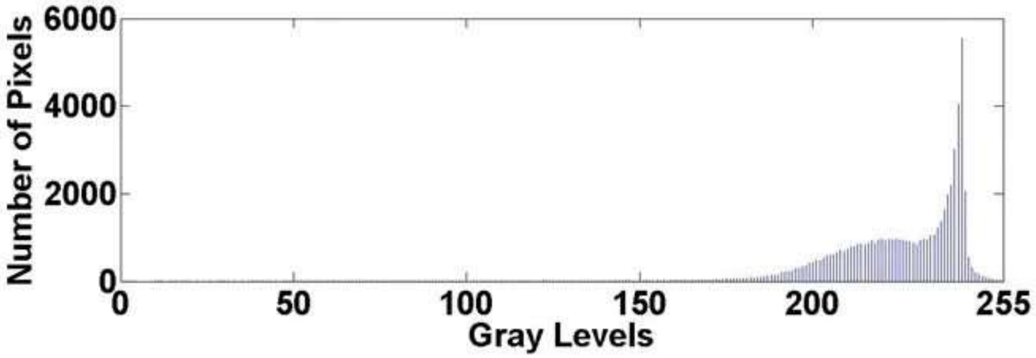

*Use ImageJ and/or FIJI Credit: Klette 2012.*

# Histogram Applications

- Check image brightness
- Check image darkness
- Histogram equalization to enhance the image
- Apply thresholding – generate binary image

# Check Image Brightness

- One can get a general idea of the brightness of an image by looking at the histogram
- If the histogram values are concentrated toward the right, the image is brighter



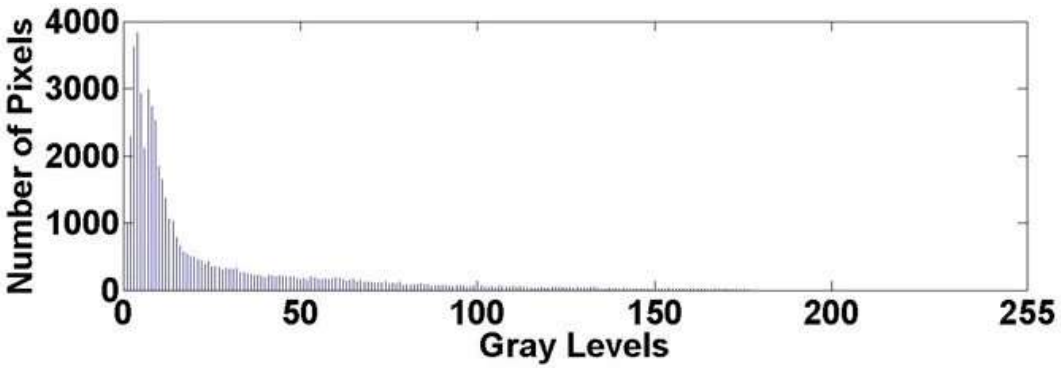

(a)

(b)

Figure credits: Sneha H.L.

# Check Image darkness

- If the histogram values are concentrated toward the left, the image is darker



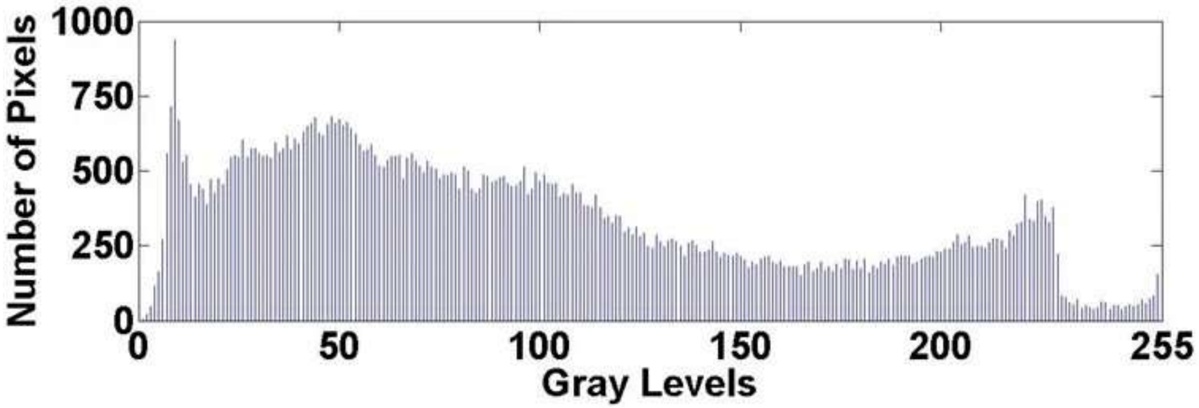

(a)

(b)

Figure credits: Sneha H.L.

# The contrast of the Image

A histogram in which the **pixel counts evenly cover a broad range of gray-scale levels** indicates an image with good/high contrast




(a)

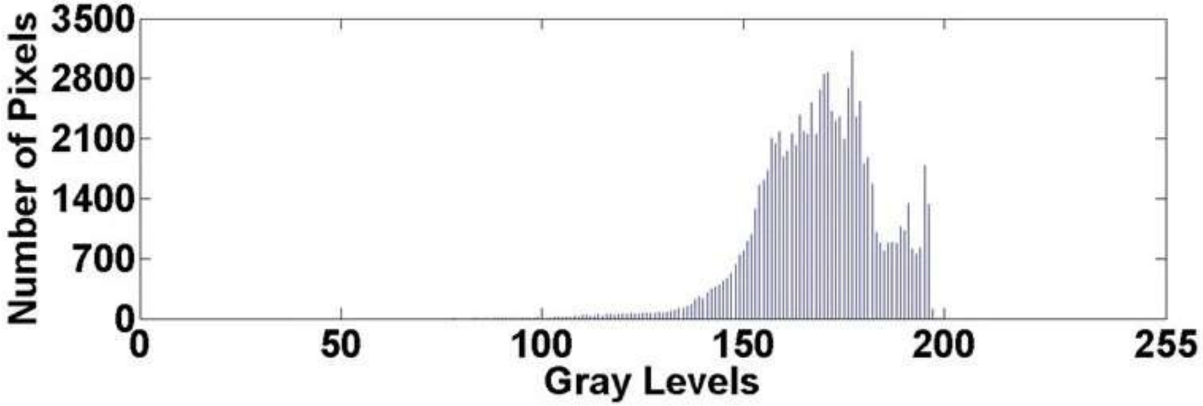
(b)

Figure 1: High contrast image

Figure credits: Sneha H.L.

# The contrast of the Image

A histogram in which the **pixel counts are not evenly cover a broad range of gray-scale levels** indicates an image with *low contrast*




(a)

(b)

Figure 2: Low contrast image

Figure credits: Sneha H.L.

# Histogram Equalization for Image Enhancement

- Enhance the quality of the image by spreading the most frequent intensity values using the following steps:

1. Calculate the frequency against each intensity value
2. Compute the probability of each frequency
3. Compute Cumulative Probability or Cumulative Distribution Function (CDF)
4. Multiply the CDF with the required intensity levels
5. Round the resultant value to floor

# Histogram Equalization for Image Enhancement

Suppose this matrix is representing pixel intensities

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

- Intensity if pixels range between 1 – 8
- Suppose we want to apply histogram equalization and scale the intensity to 1 - 20

# Histogram Equalization for Image Enhancement

**Example**

1. Calculate the frequency against each intensity value

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

| Intensity | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Count | 1 | 3 | 3 | 2 | 2 | 1 | 3 | 1 |

# Histogram Equalization for Image Enhancement

**Example**

2. Compute the probability of each frequency

$$probability = \frac{\text{No. of pixels associated to a particular intensity}}{\text{Total pixels}}$$

# Histogram Equalization for Image Enhancement

**Example**

2. Compute the probability of each frequency

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

| Intensity | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Count | 1 | 3 | 3 | 2 | 2 | 1 | 3 | 1 |
| Probability | .062 | .187 | .187 | .125 | .125 | .062 | .187 | .062 |

# Histogram Equalization for Image Enhancement

**Example**

3. Compute Cumulative Distribution Function (CDF) i.e. cumulative probabilities

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

| Intensity | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Count | 1 | 3 | 3 | 2 | 2 | 1 | 3 | 1 |
| Probability | .0625 | .1875 | .1875 | .125 | .125 | .0625 | .1875 | .0625 |
| Cumulative Probability | .0625 | .25 | .4375 | .5625 | .6875 | .75 | .9375 | 1 |

# Histogram Equalization for Image Enhancement

## Example

4. Multiply the CDF with the required intensity levels
   - As we want to increase the intensity to the range 1-20, multiply each cumulative probability by 20

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

| **Intensity** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|
| **Count** | **1** | **3** | **3** | **2** | **2** | **1** | **3** | **1** |
| **Probability** | **.0625** | **.1875** | **.1875** | **.125** | **.125** | **.0625** | **.1875** | **.0625** |
| **Cumulative Probability** | **.0625** | **.25** | **.4375** | **.5625** | **.6875** | **.75** | **.9375** | **1** |
| **CP x 20** | **1.25** | **5** | **8.75** | **11.25** | **13.75** | **15** | **18.75** | **20** |

# Histogram Equalization for Image Enhancement

Example

5. Round the resultant value to the floor

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

| Intensity | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Count | 1 | 3 | 3 | 2 | 2 | 1 | 3 | 1 |
| Probability | .0625 | .1875 | .1875 | .125 | .125 | .0625 | .1875 | .0625 |
| Cumulative Probability | .0625 | .25 | .4375 | .5625 | .6875 | .75 | .9375 | 1 |
| CP x 20 | 1.25 | 5 | 8.75 | 11.25 | 13.75 | 15 | 18.75 | 20 |
| Floor Rounding | 1 | 5 | 8 | 11 | 13 | 15 | 18 | 20 |

# **Histogram** Equalization for Image Enhancement

**Example**

Original Image has been transformed to equalized image with different intensities

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

**Before**

| | | | |
|---|---|---|---|
| 8 | 5 | 11 | 13 |
| 18 | 18 | 20 | 5 |
| 8 | 1 | 5 | 8 |
| 13 | 11 | 15 | 18 |

**After**

# Python function for Histogram Equalization

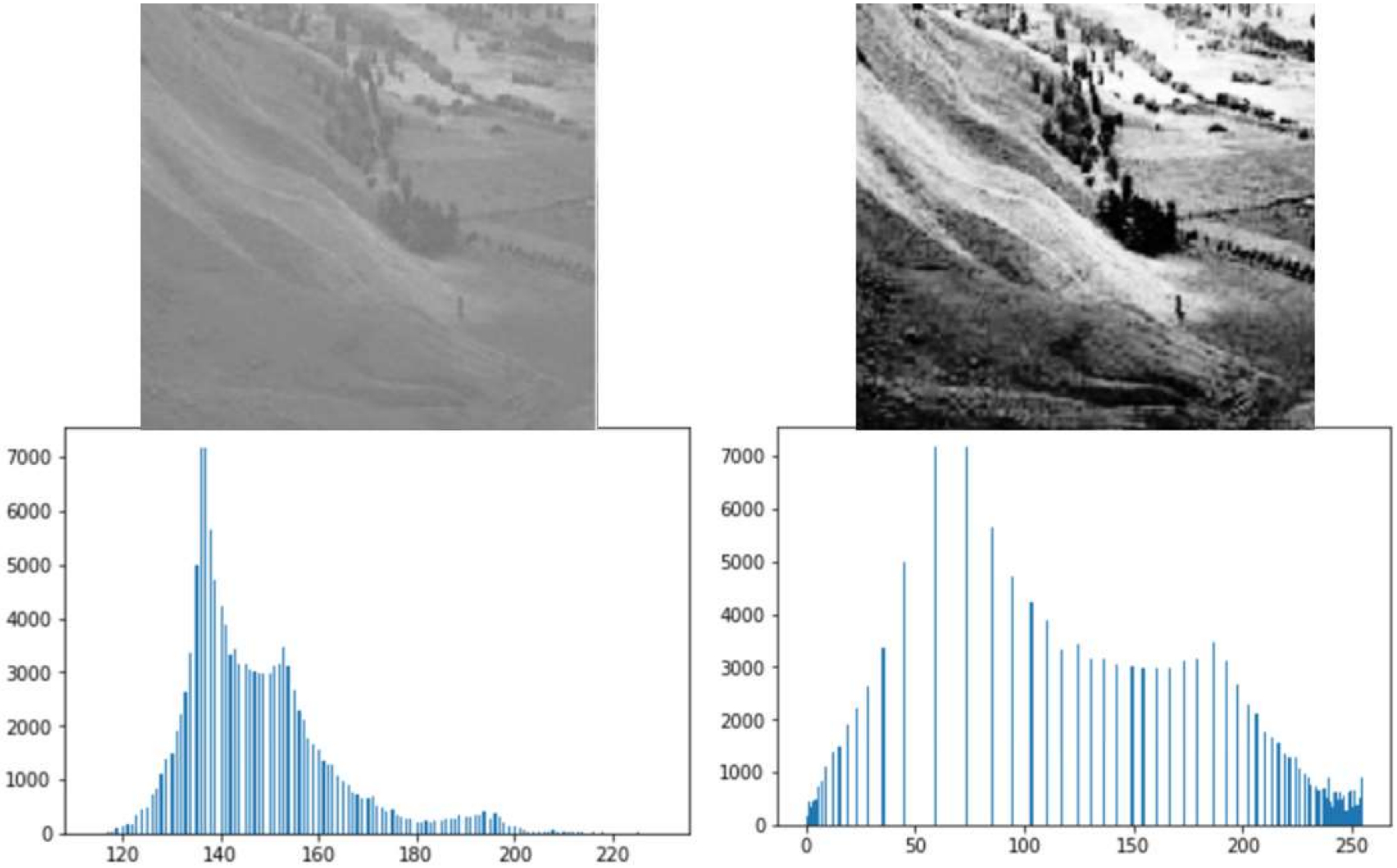
```
# Importing libraries
import cv2 as cv

# OpenCV function
cv.equalizeHist(image)
```

```
#  Importing libraries
import   cv2
import   matplotlib.pyplot as plt
def histogram_equalize(img):
    b, g, r = cv2.split(img)
    red = cv2.equalizeHist(r)
    green = cv2.equalizeHist(g)
    blue = cv2.equalizeHist(b)
    return cv2.merge((blue, green, red))

img = cv2.imread('eq.jpg')
plt.imshow(img)
img.shape

equ = histogram_equalize(img)
equ.shape
res = np.hstack((img,equ)) #stacking images side-by-side
plt.imshow(res)
```



7000
6000
5000
4000
3000
2000
1000
0
120
140
160
180
200
220
7000
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250

# Take-Home Quiz 1

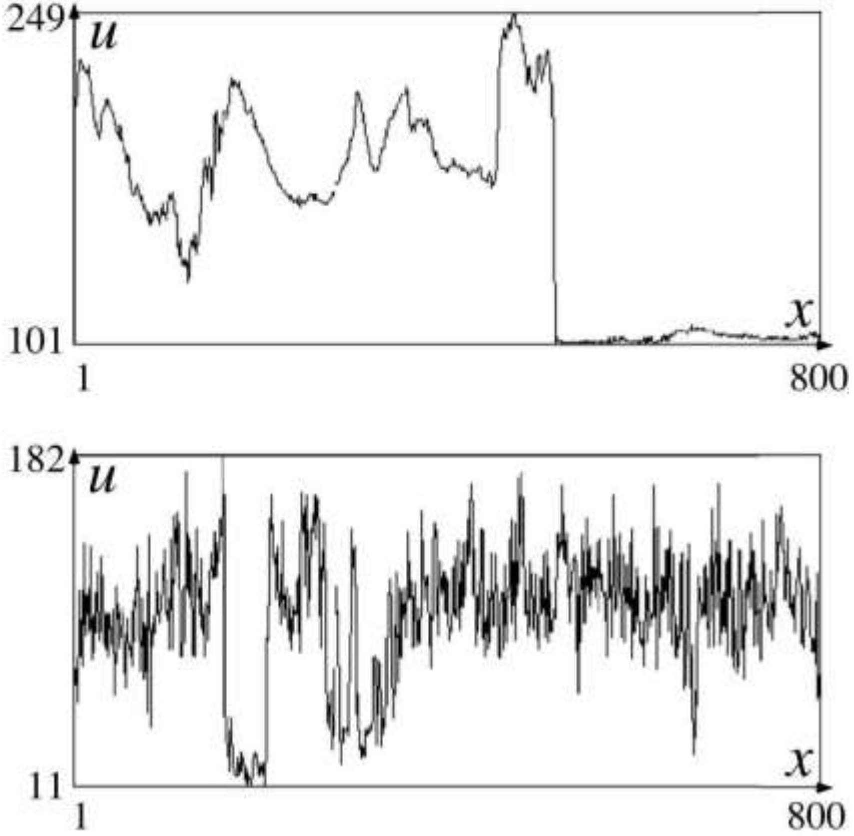
a. Read and convert color(RGB) image to gray-scale image

$$f'(x, y) = g(f(x, y))$$

where $f'(x, y)$ is the gray-scale image and f in representing a color image, whereas g is a function that is used to convert the color image to gray-scale

b. Apply the provide code (shown in previous slides ) to enhance contrast of any low contrast image (dark or bright)

# Intensity profiles for selected (two) rows



# Image noise

- Light Variations
- Camera Electronics
- Surface Reflectance
- Lens

- Noise is random,
  - it occurs with some probability
- It has a distribution

# Noise

- $I_{original}(x,y)$ – true pixel value at (x,y)
- n(x,y) - noise at (x,y)
- $I_{observed}(x,y) = I_{original}(x,y) + n(x,y)$ additive noise

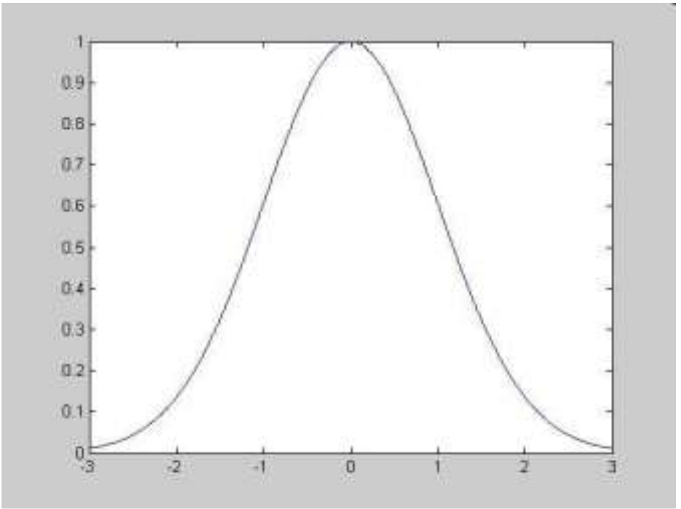
# Noise

- $I_{original}(x,y)$ – true pixel value at (x,y)
- $n(x,y)$ - noise at (x,y)
- $I_{observed}(x,y) = I_{original}(x,y) * n(x,y)$ multiplicative noise

# Gaussian Noise

$$n(x,y) \approx g(n) = e^{\frac{-n^2}{2\sigma^2}}$$

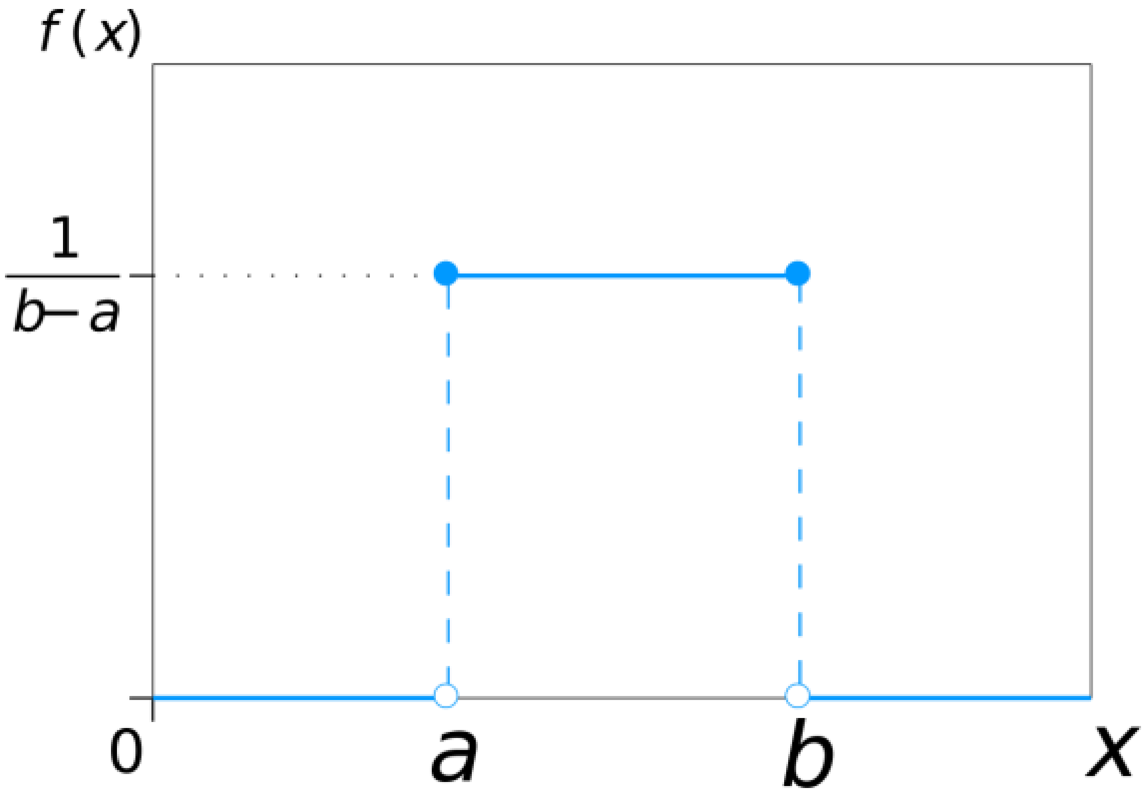
$g(n)$




$n$

Probability Distribution
*n* is a random variable

# Uniform distribution



# Salt and pepper noise

- Each pixel is randomly made black or white with a uniform probability distribution

➔

Salt-pepper

# So…. How do we detect an object in an image?

# Naïve approach: Template Matching

This is a chair

Find the chair in this image

Output of correlation

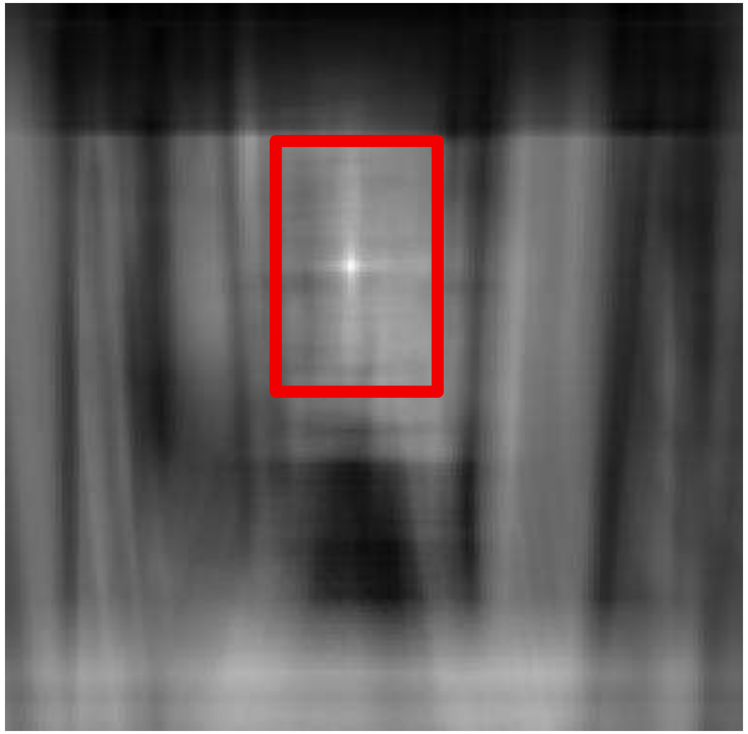




Image source: Antonio Torralba

# Template Matching

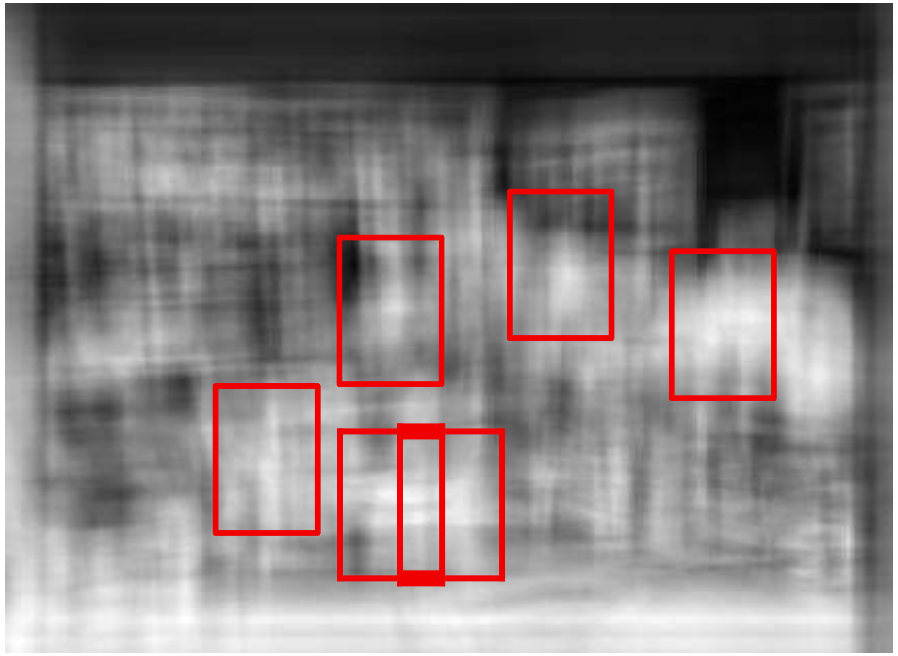
Find the chair in this image



Image source: Antonio Torralba

# Template Matching

Find the chair in this image



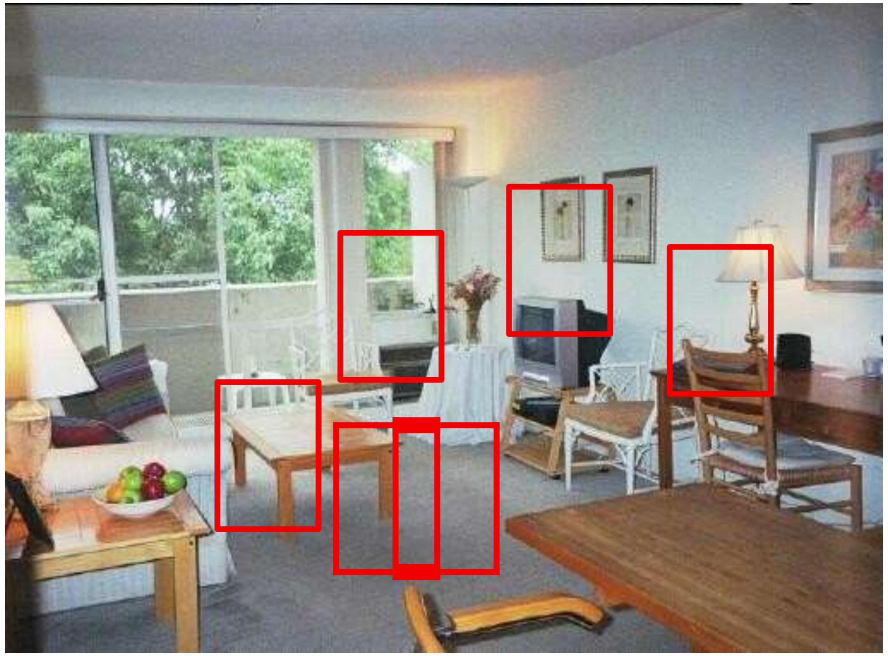
Epic fail!

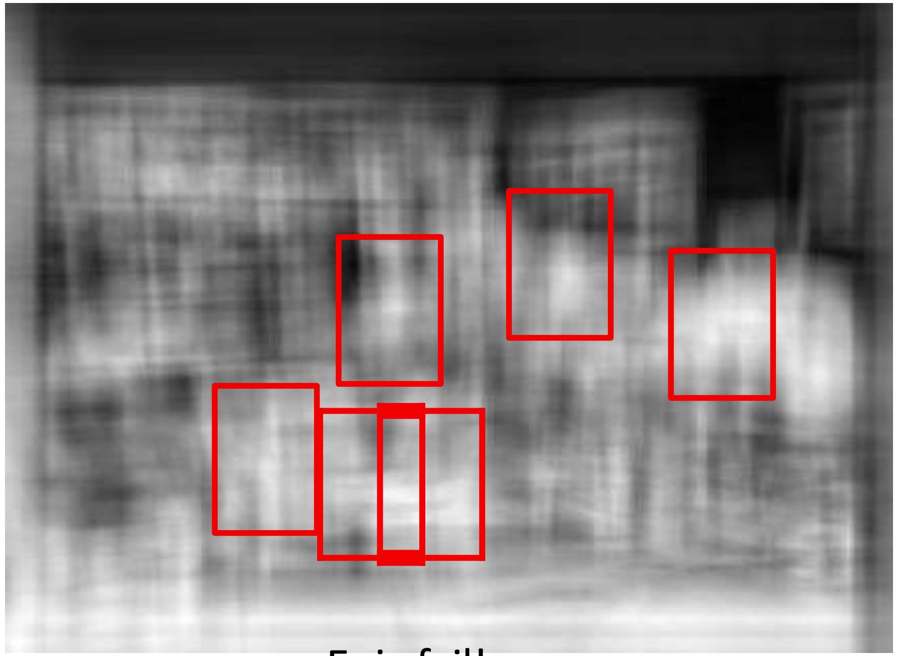
Simple template matching is not going to make it



Image source: Antonio Torralba

# Template Matching





Epic fail!

Simple template matching is not going to make it

Find the chair in this image



Image source: Antonio Torralba

# Idea

- Instead of comparing raw image pixels:
  - First map those pixels into another (more robust) form,
  - And then compare those mapped forms.
  - Finally, select the closest image map (how do you define “closest”? Metrics).
- Features, examples:
  - compute edges
  - compute color histograms
  - gradients
  - HOG
  - SIFT
  - …

# Image filtering

- Image filtering: compute function of local neighborhood at each position

h=output f=filter I=image

$$h[m,n] = \sum_{k,l} f[k,l]\, I[m+k, n+l]$$

2d coords=k,l 2d coords=m,n

[ ] [ ] [ ]

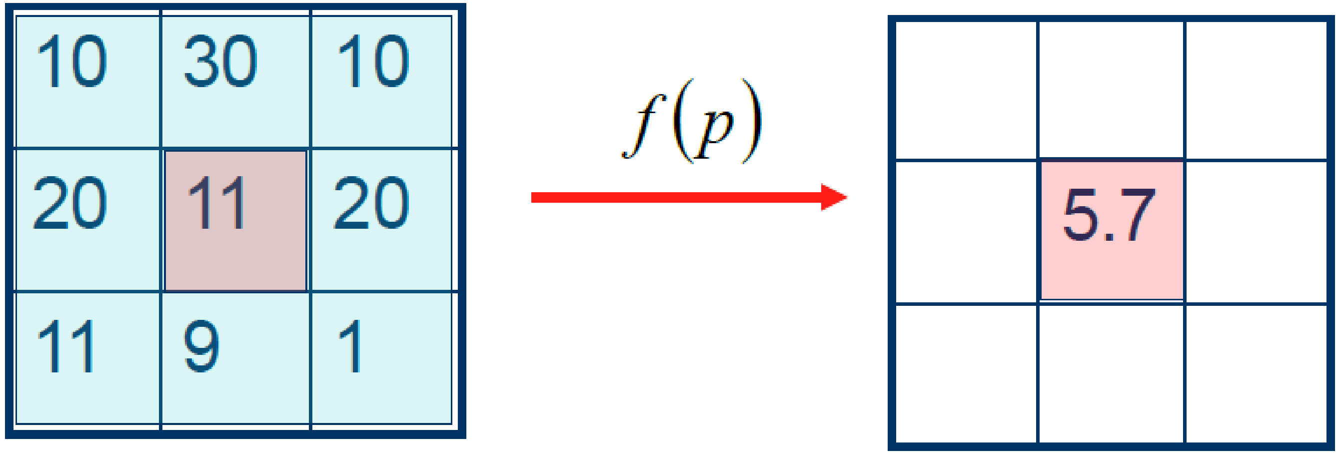
# Image filtering

- Image filtering: compute function of local neighborhood at each position

- Enhance images
  - Denoise, resize, increase contrast, etc.
- Extract information from images
  - Texture, edges, distinctive points, etc.
- Detect patterns
  - Template matching

# Filtering

- Modify pixels based on some function of neighborhood



# Filtering

- Output is linear combination of the neighborhood pixels

$$\underset{\text{Image}}{\begin{bmatrix}1 & 3 & 0\\ 2 & 10 & 2\\ 4 & 1 & 1\end{bmatrix}} \otimes \underset{\text{Kernel}}{\begin{bmatrix}1 & 0 & -1\\ 1 & 0.1 & -1\\ 1 & 0 & -1\end{bmatrix}} = \underset{\text{Filter Output}}{\begin{bmatrix} & & \\ & 5 & \\ & & \end{bmatrix}}$$

# Derivatives and Average

- Derivative: rate of change
  - Speed is a rate of change of a distance, X=V.t
  - Acceleration is a rate of change of speed, V=a.t
- Average: mean
  - Dividing the sum of N values by N

# Derivative

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x) = f_x$$

$$y = x^2 + x^4$$

$$\frac{dy}{dx} = 2x + 4x^3$$

$$y = \sin x + e^{-x}$$

$$\frac{dy}{dx} = \cos x + (-1)e^{-x}$$

# Discrete Derivative

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x)$$

$$\frac{df}{dx} = \frac{f(x) - f(x-1)}{1} = f'(x)$$

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

# Discrete Derivative / Finite Difference

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

Backward difference

$$\frac{df}{dx} = f(x) - f(x+1) = f'(x)$$

Forward difference

$$\frac{df}{dx} = f(x+1) - f(x-1) = f'(x)$$

Central difference

# Example: Finite Difference

$$f(x) = 10 \quad 15 \quad 10 \quad 10 \quad 25 \quad 20 \quad 20 \quad 20$$

$$f'(x) = \ 0 \quad 5 \quad -5 \quad 0 \quad 15 \ -5 \quad 0 \quad 0$$

$$f''(x) = \ 0 \quad 5 \quad 10 \quad 5 \quad 15 \ -20 \ 5 \quad 0$$

## Derivative Masks

| | |
|---|---|
| Backward difference | [-1 1] |
| Forward difference | [1 -1] |
| Central difference | [-1 0 1] |

# Derivative in 2-D

Given function $f(x,$

Gradient vector $$\nabla f(x,y) = \begin{bmatrix} \dfrac{\partial f(x,y)}{\partial x} \\ \dfrac{\partial f(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

Gradient magnitude $$\left|\nabla f(x,y)\right| = \sqrt{\quad}$$

Gradient direction $$\theta = \tan^{-1} \frac{f_x}{f_y}$$

# Derivative of Images

Derivative masks $f_x \Rightarrow \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ $f_y \Rightarrow \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \boxed{10} & \boxed{10} & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

# Derivative of Images

Derivative masks $f_x \Rightarrow \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ $f_y \Rightarrow \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

# Averages

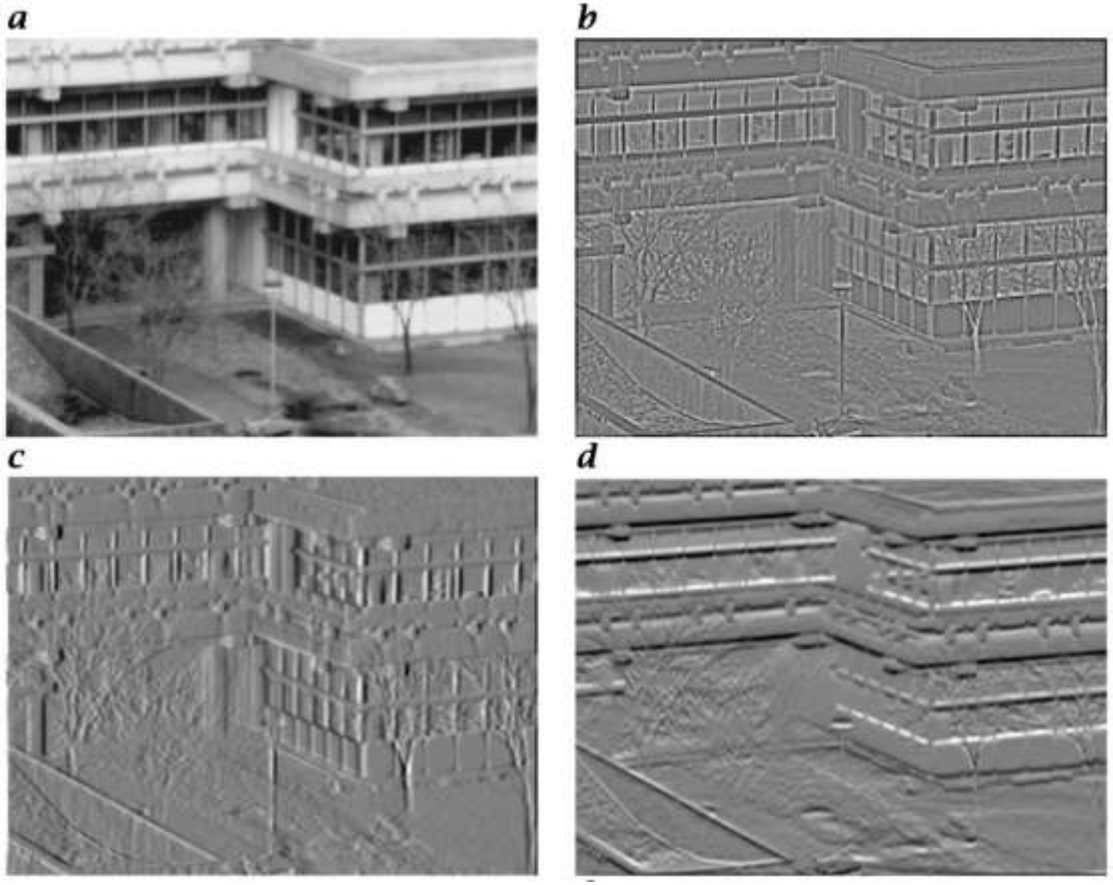
- Mean

$$I = \frac{I_1 + I_2 + \dots I_n}{n} = \frac{\sum_{i=1}^{n} I_i}{n}$$

- Weighted mean

$$I = \frac{w_1 I_1 + w_2 I_2 + \dots + w_n I_n}{n} = \frac{\sum_{i=1}^{n} w_i I_i}{n}$$

# Example




a. Original image
b. Laplacian operator
c. Horizontal derivative
d. Vertical derivative

# Correlation (linear relationship)

$$f \otimes h = \sum \sum f(k$$

$$f = \text{Image}$$

$$h = \text{Kernel}$$

$$
\underset{f}{\begin{array}{|c|c|c|}
\hline
f_1 & f_2 & f_3 \\
\hline
f_4 & f_5 & f_6 \\
\hline
f_7 & f_8 & f_9 \\
\hline
\end{array}}
\otimes
\underset{h}{\begin{array}{|c|c|c|}
\hline
h_1 & h_2 & h_3 \\
\hline
h_4 & h_5 & h_6 \\
\hline
h_7 & h_8 & h_9 \\
\hline
\end{array}}
\longrightarrow
\begin{aligned}
f \otimes h = {} & f_1 h_1 + f_2 h_2 + f_3 h_3 \\
& + f_4 h_4 + f_5 h_5 + f_6 h_6 \\
& + f_7 h_7 + f_8 h_8 + f_9 h_9
\end{aligned}
$$

# Convolution

$$f * h = \sum \sum f\left(k, l\right.$$

$f$ = Image

$h$ = Kernel

$h$

$$\begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \xrightarrow{X - flip} \begin{bmatrix} h_7 & h_8 & h_9 \\ h_4 & h_5 & h_6 \\ h_1 & h_2 & h_3 \end{bmatrix} \xrightarrow{Y - flip} \begin{bmatrix} h_9 & h_8 & h_7 \\ h_6 & h_5 & h_4 \\ h_3 & h_2 & h_1 \end{bmatrix}$$

$f$

$$\begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} * \begin{bmatrix} h_9 & h_8 & h_7 \\ h_6 & h_5 & h_4 \\ h_3 & h_2 & h_1 \end{bmatrix} \longrightarrow$$

$$\begin{aligned} f * h = {} & f_1 h_9 + f_2 h_8 + f_3 h_7 \\ & + f_4 h_6 + f_5 h_5 + f_6 h_4 \\ & + f_7 h_3 + f_8 h_2 + f_9 h_1 \end{aligned}$$

# Convolution

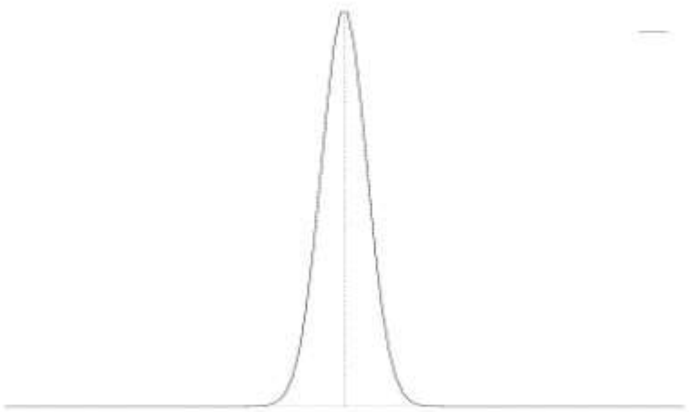
- Convolution is associative

$$F * (G * I) = ($$

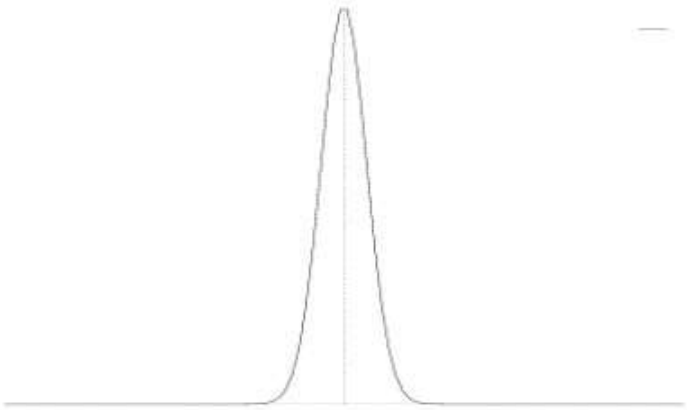
# Correlation and Convolution

- Convolution is a filtering operation
  - expresses the amount of overlap of one function as it is shifted over another function
- Correlation compares the similarity of two sets of data
  - relatedness of the signals!

# Gaussian filter

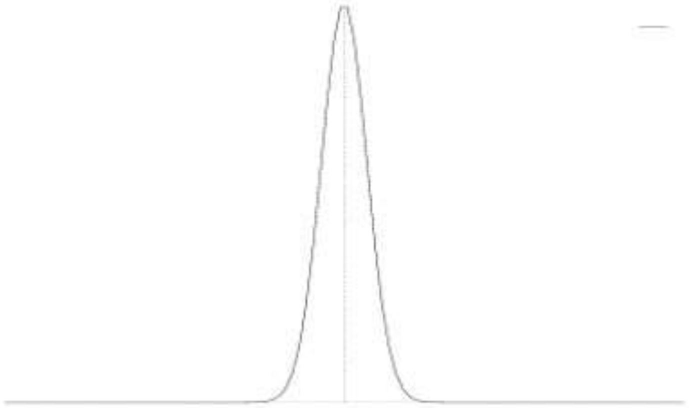


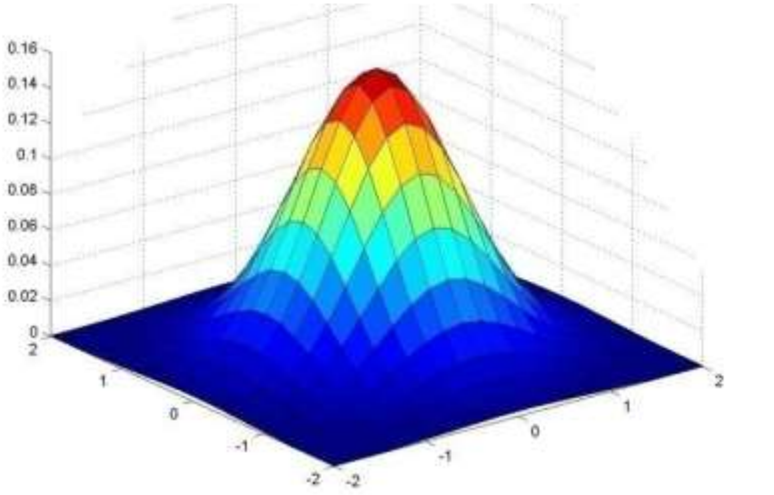
# Gaussian filter



$$g(x) = e^{\frac{-x^2}{2\sigma^2}}$$

# Gaussian filter

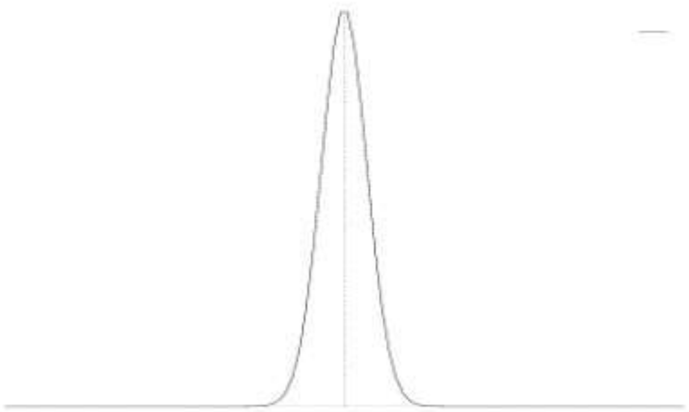


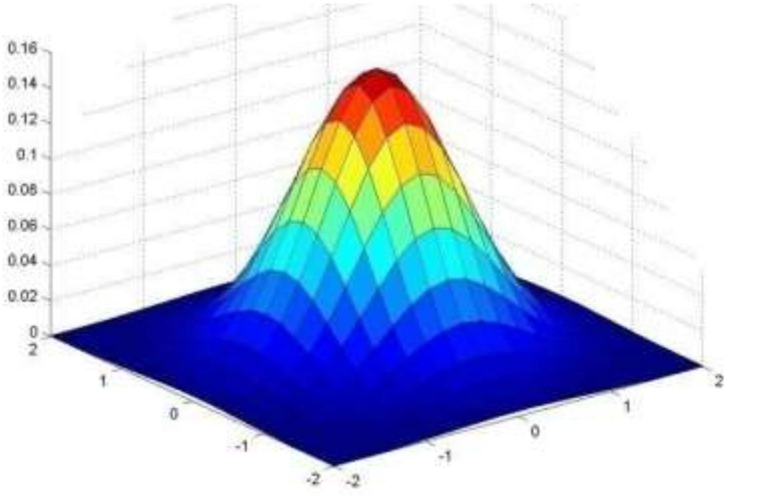



$$g(x) = e^{\frac{-x^2}{2\sigma^2}}$$

$$g(x, y) = e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

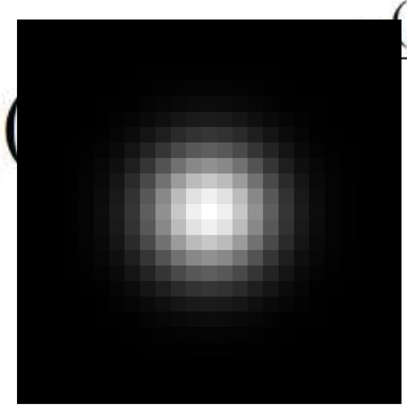
# Gaussian filter



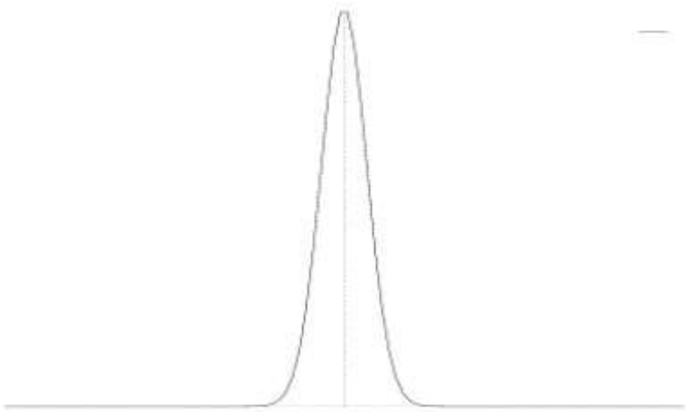


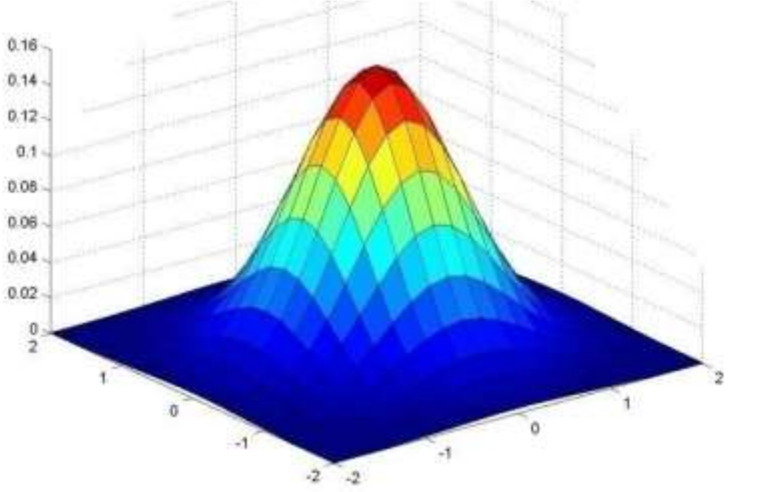

$$g(x) = e^{\frac{-x^2}{2o^2}}$$

$$g( \quad \frac{(x^2+y^2)}{2o^2}$$



# Gaussian filter

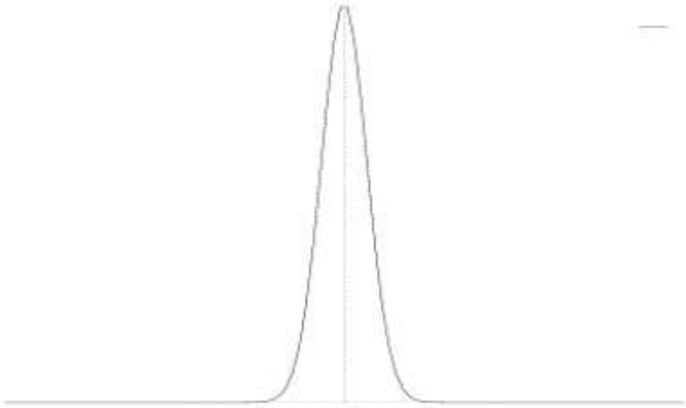


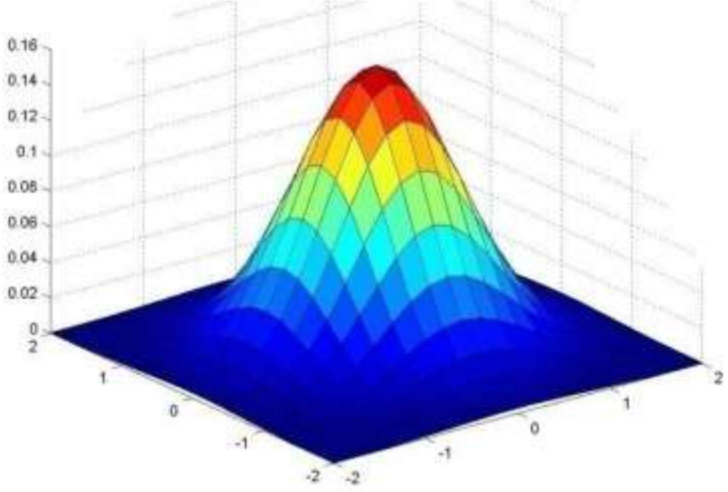



$$g(x) = e^{\frac{-x^2}{2\sigma^2}}$$

$$g(x, y) = e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

# Gaussian filter





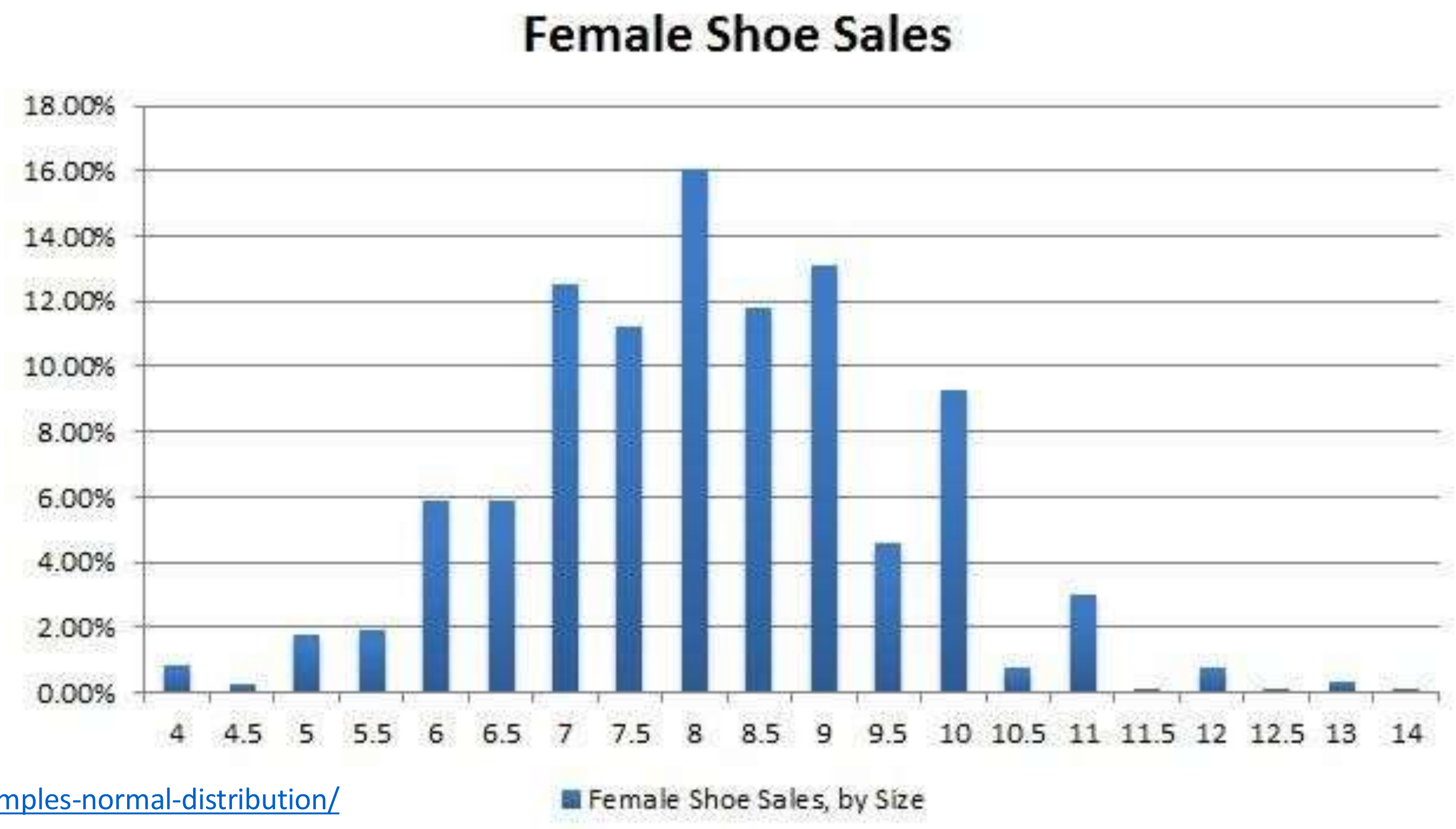

$$g(x) = e^{\frac{-x^2}{2\sigma^2}}$$

$$g(x,y) = e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

$$g(x) = [.011 \quad .13 \quad .6 \quad 1 \quad .6 \quad .13 \quad .011]$$

# Gaussian filter - properties

- Most common natural model

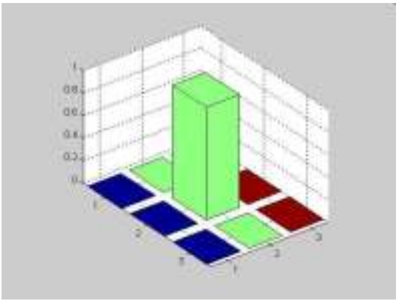



https://studiousguy.com/real-life-examples-normal-distribution/

# Gaussian filter - properties

- Most common natural model
- Smooth function, it has infinite number of derivatives
- It is Symmetric
- Fourier Transform of Gaussian is Gaussian.
- Convolution of a Gaussian with itself is a Gaussian.
- Gaussian is separable; 2D convolution can be performed by two 1-D convolutions
- There are cells in eye that perform Gaussian filtering.

# Filtering Examples - 1

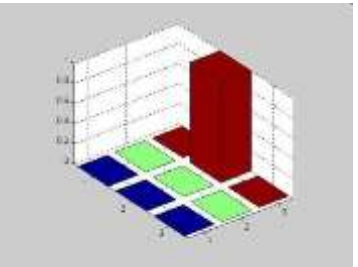


*

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

# Filtering Examples - 2

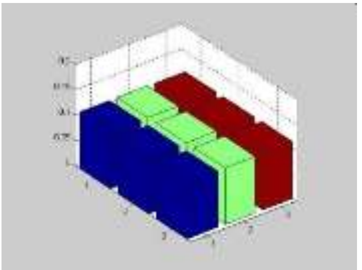
$*$



| | | |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 0 |

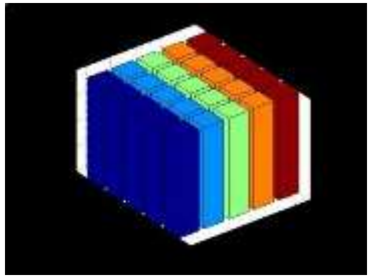
$=$

# Filtering Examples - 3



$$* \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} =$$

# Filtering Examples - 4



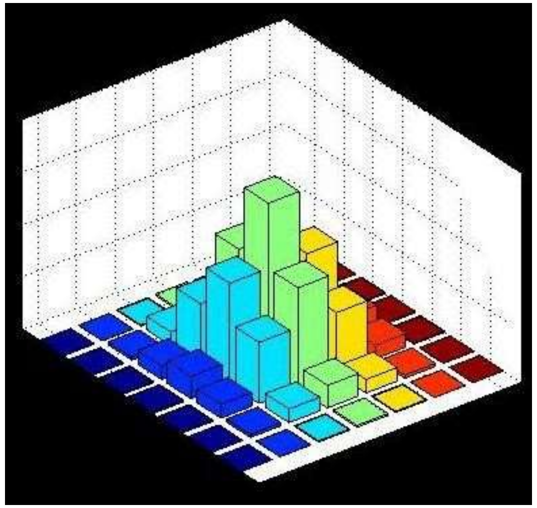
$$* \frac{1}{25}$$

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

$$=$$

# Filtering Examples - 5

*



=

*Gaussian Smoothing*

# Filtering Examples - 6

Gaussian Smoothing

Smoothing by Averaging

# Filtering Examples - 7

After additive Gaussian Noise

After Averaging

After Gaussian Smoothing

# Questions?

Sources for this lecture include materials from works by Mubarak Shah, S. Seitz, James Tompkin and Ulas Bagci