LECTURE 02 IMAGE FORMATION, COLORS, ARITHMETIC OPERATIONS

Punnarai Siricharoen, Ph.D.

OBJECTIVES

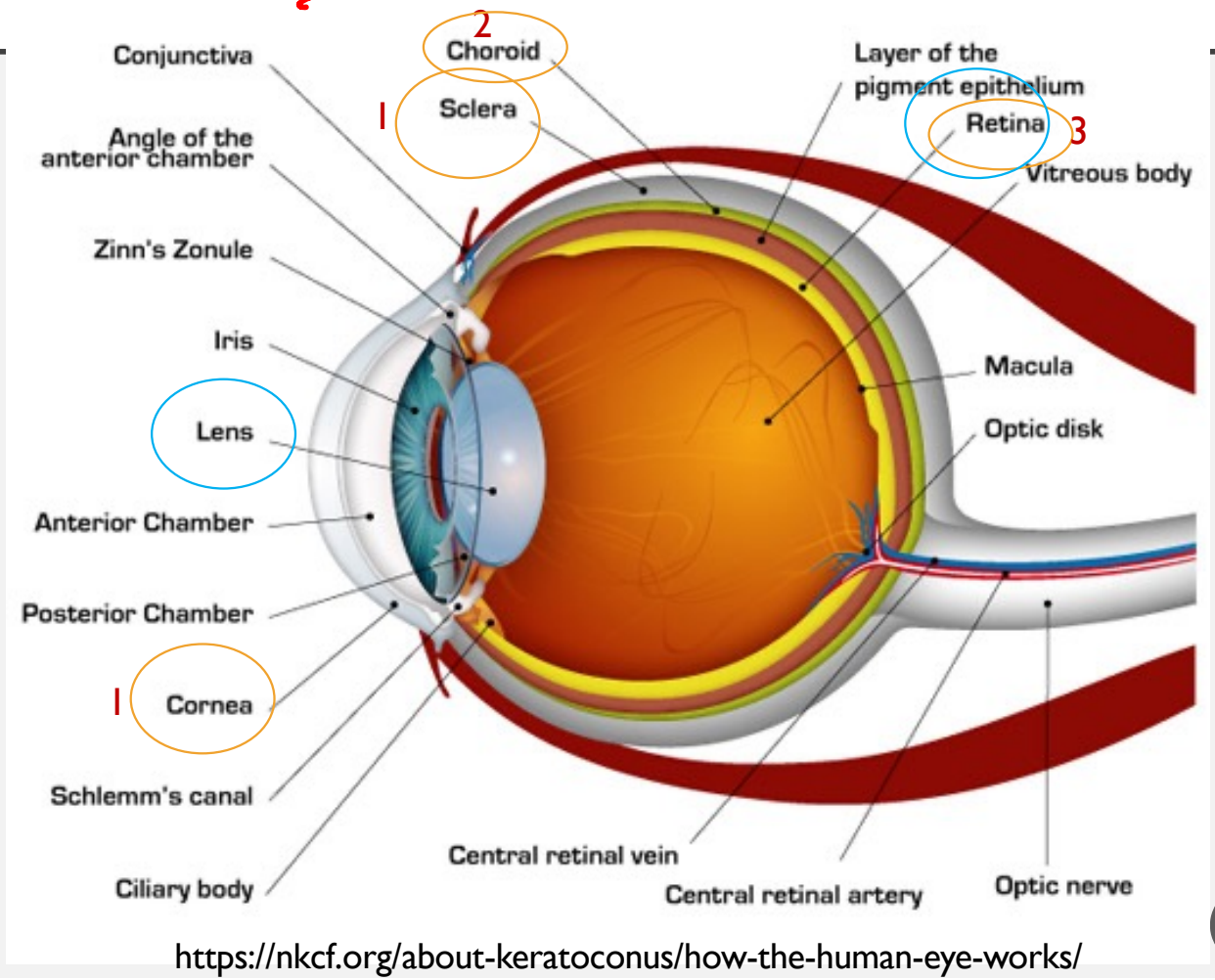
- To learn how an image is formed.
- To understand different color models and their purposes.
- To be able to use arithmetic operations for processing an image.

TODAY'S CONTENT

- Image formation
- Color models
- Arithmetic Operations
- Exercises

HUMAN EYE

- Lens: 60-70% water, 6% fat, high protein (controlled flexibility by ciliary body)
- Retina: properly focused, light imaged to retina plane
  - Two classes of receptors : sensor
  1. Cones
  2. Rods



https://nkcf.org/about-keratoconus/how-the-human-eye-works/

IMAGE FORMATION IN HUMAN EYE

ptstvibutfl on

- **Cones:** 6-7 millions,
  - sensitive to color + fine details
  - central of the retina (fovea)
  - Resolve fine details

- **Rods:** 75-150 millions
  - general, overall picture
  - distribute all over the retina
  - Sensitive to low level of illumination
  - Not involved in color vision

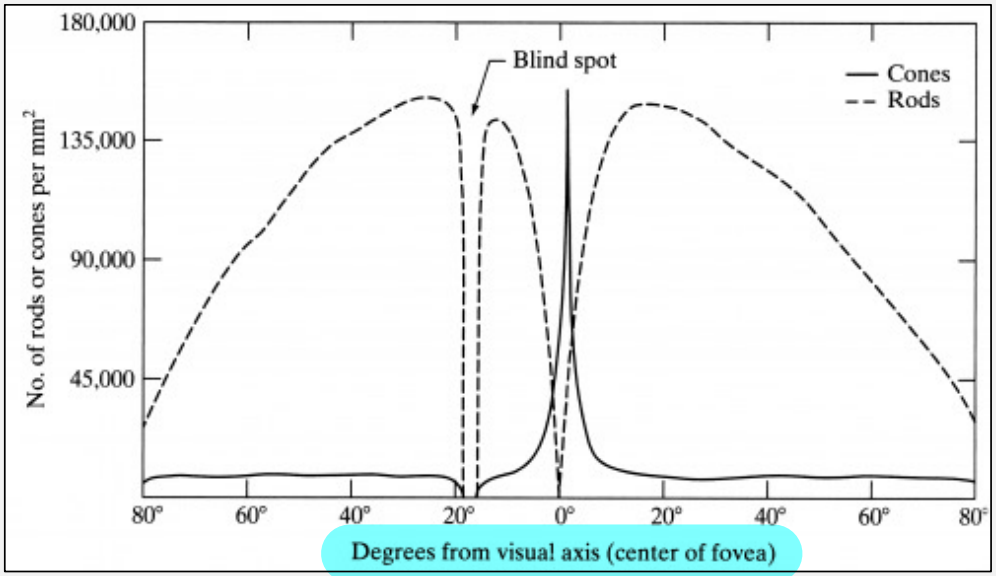
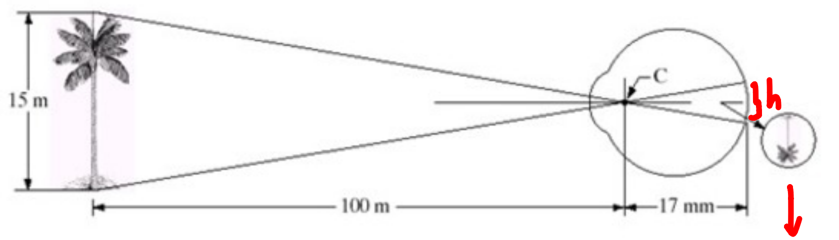


Image from Gonzalez & Woods, Digital Image Processing

IMAGE FORMATION IN HUMAN EYE

FIGURE 2.3 Graphical representation of the eye looking at a palm tree. Point *C* is the optical center of the lens.



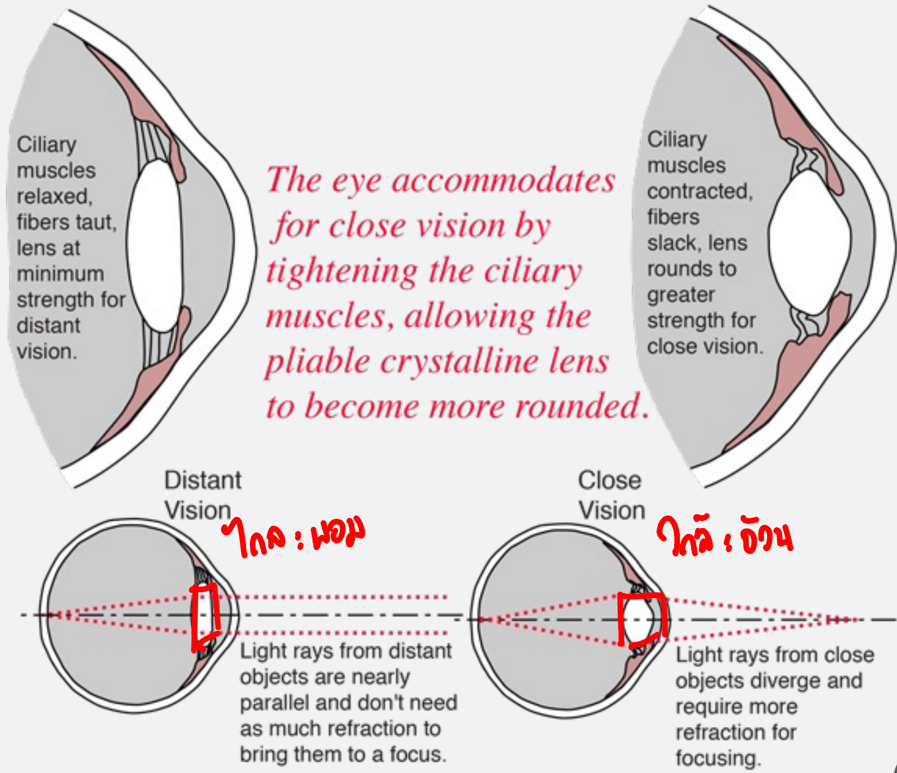
© Gonzalez & Woods, Digital Image Processing

$$\frac{h}{17} = \frac{15}{100}$$

$$h = 2.55\,mm$$

↓ ขนาดรูปบน retina

How the focal lens change by distance:



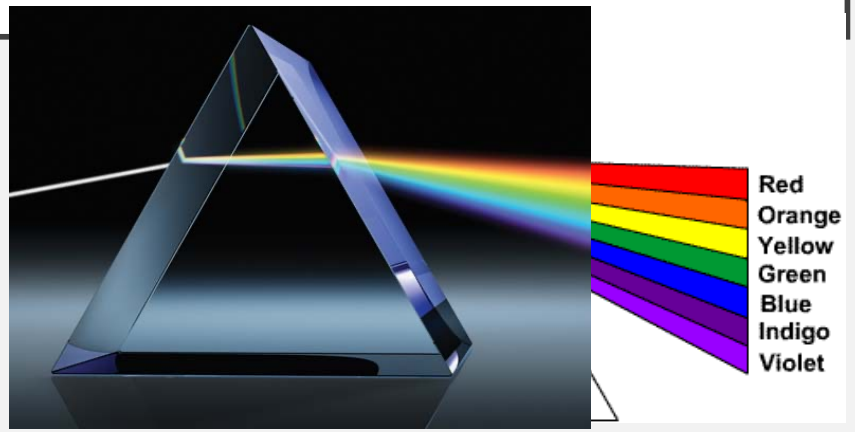
http://hyperphysics.phy-astr.gsu.edu/hbase/vision/accom.html

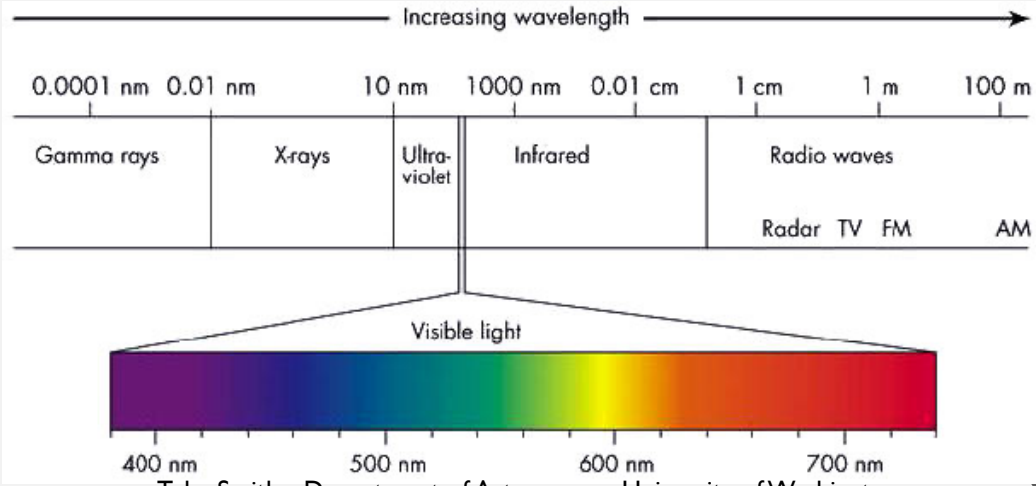
LIGHT AND ELECTROMAGNETIC SPECTRUM

- Sir Isaac Newton found the beam of white light consists of a continuous spectrum.
- A range of colour -> electromagnetic spectrum

$$\lambda = \frac{c}{v}$$

- λ - Wavelength
- c – speed of light
- v - frequency



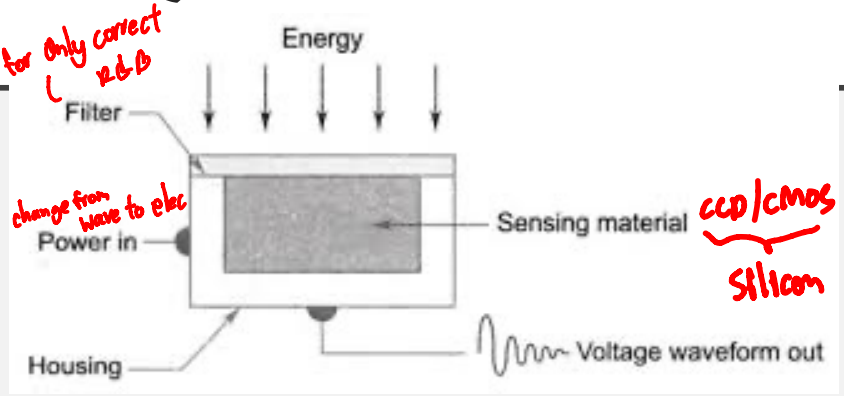


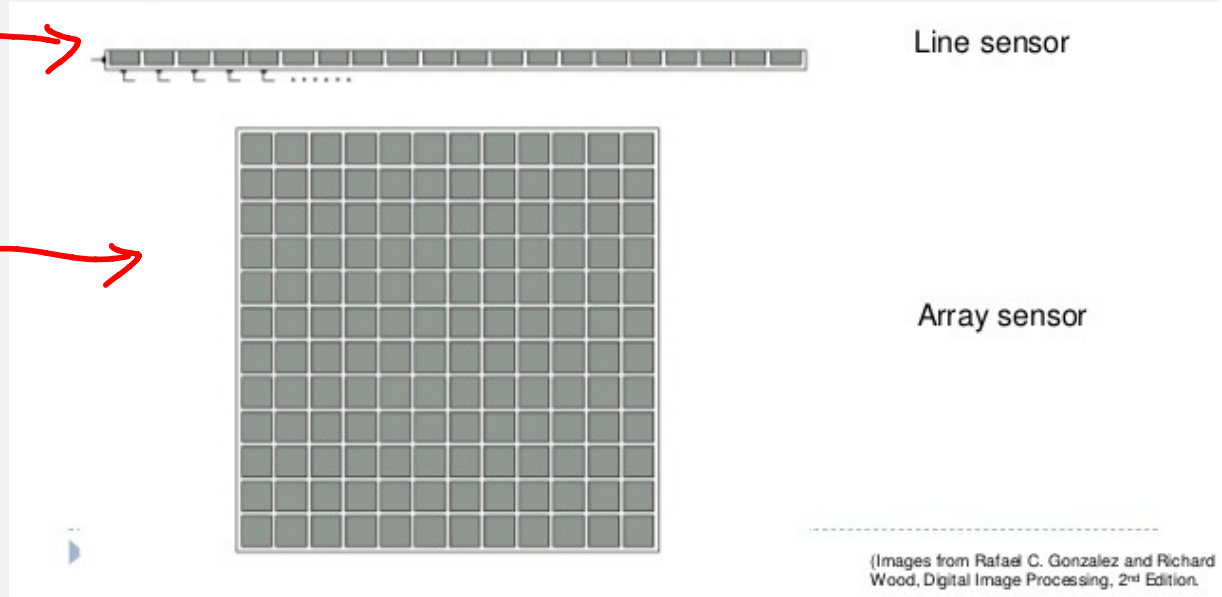
Toby Smith - Department of Astronomy - University of Washington

IMAGE SENSING AND ACQUISITION

Three sensor arrangements:

- Single Imaging Sensor
  - Require motions to create 2D image
- Line Sensor
  - Require motions to create 2D image
  - Ring configurations -> medical app
- Array Sensor
  - Electromagnetic & some ultrasound sensing devices, digital camera (CCD array)
  - Low-noise image application (astronomical app)





(Images from Rafael C. Gonzalez and Richard Wood, Digital Image Processing, 2nd Edition.

IMAGE SENSING AND ACQUISITION

- Image Acquisition Process
  - Image function characterized by
    - (1) Amount of source illumination incident on the scene being viewed (illumination)
    - (2) Amount of illumination reflected by the object in the scene (reflectance)

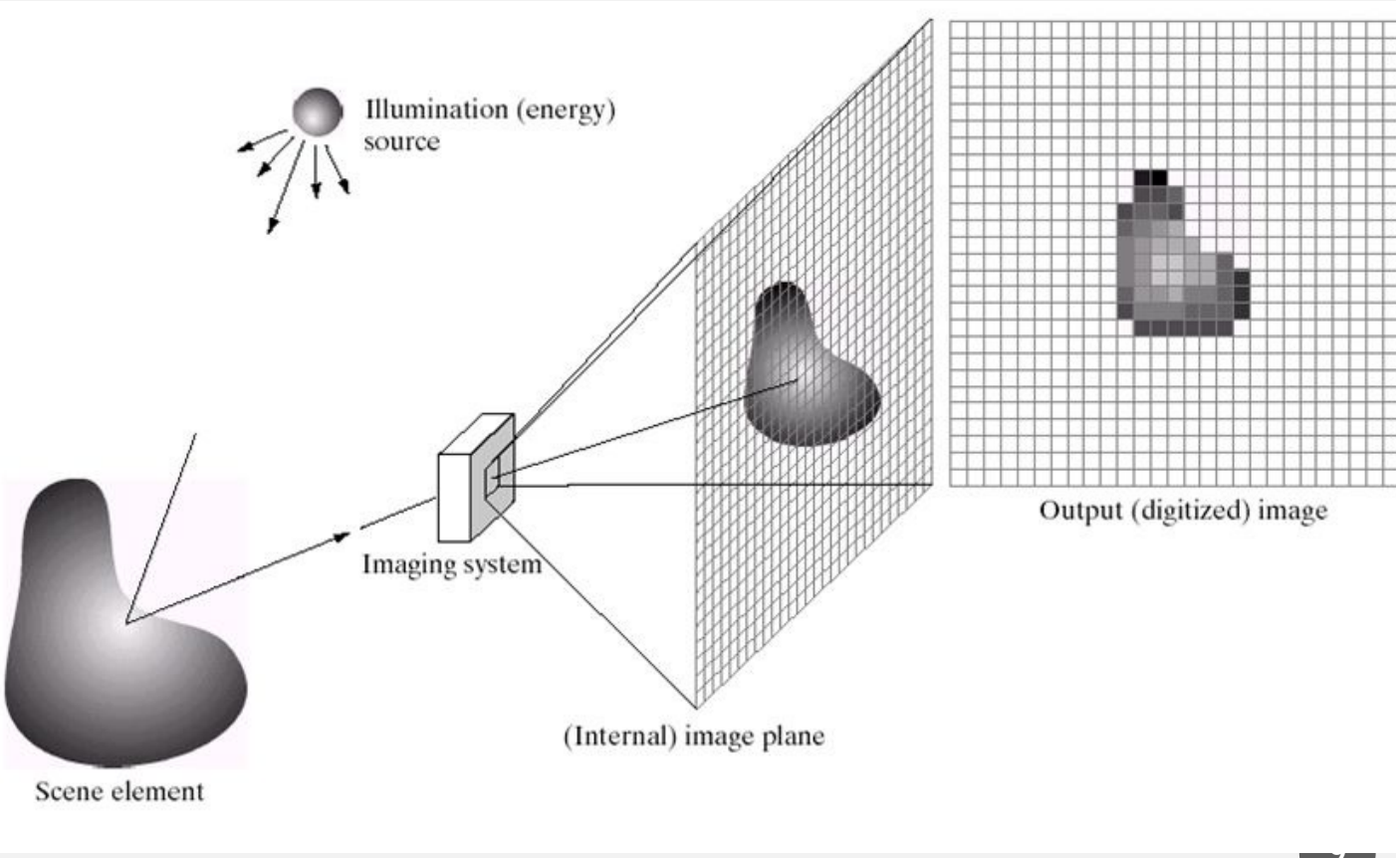
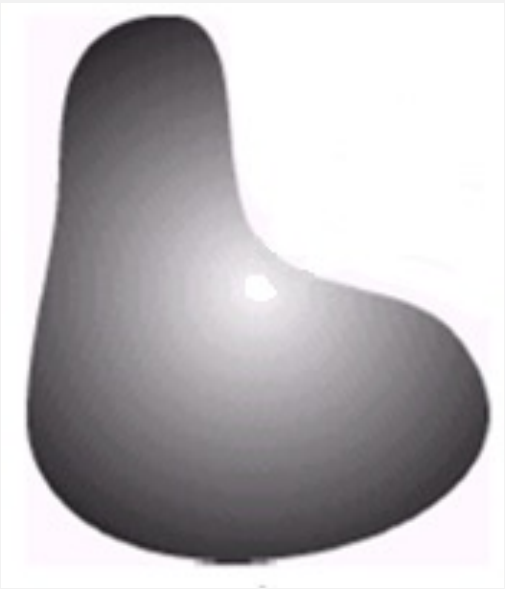


Image from Gonzalez & Woods, Digital Image Processing

HOW?



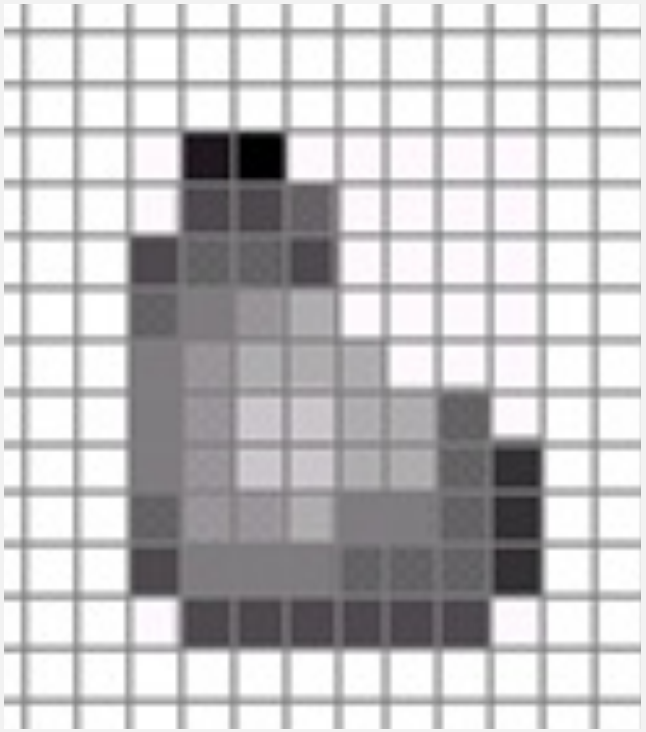


IMAGE SAMPLING AND QUANTIZATION

- Convert the continuous sensed data into a digital form:
  - Sampling & Quantization

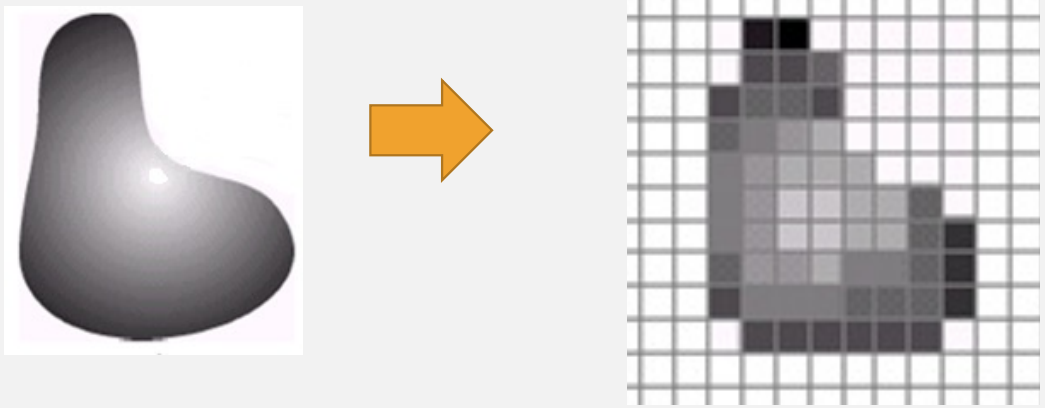
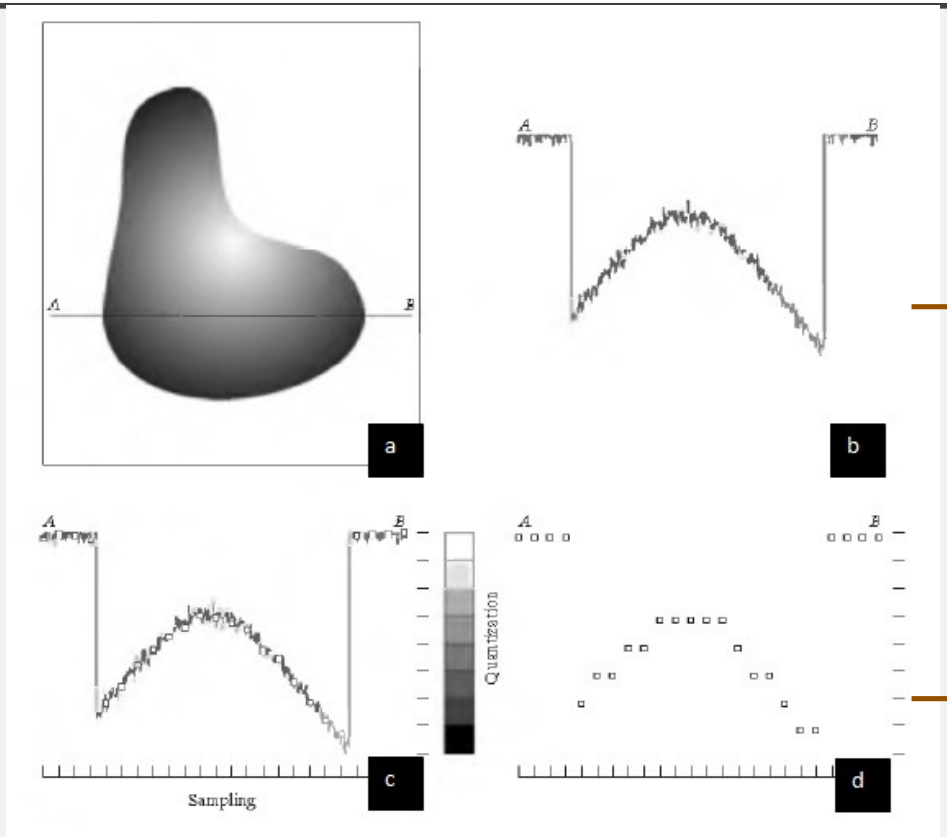


IMAGE SAMPLING AND QUANTIZATION

- **Sampling** digitizing the coordinate values.

- **Quantization** digitizing the amplitude values.



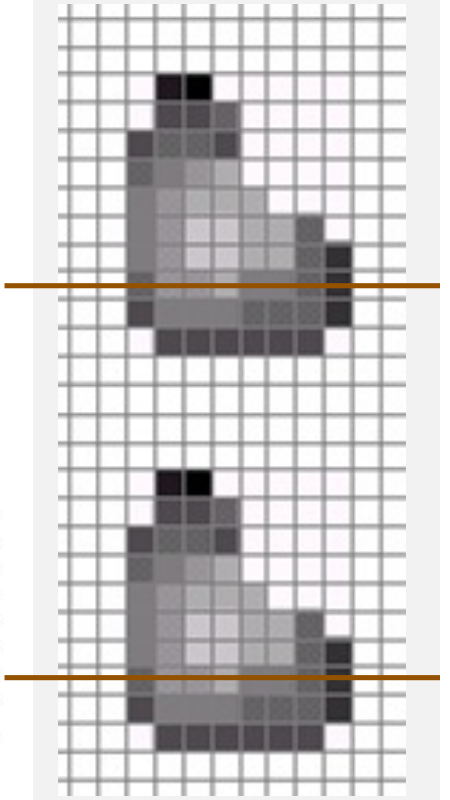
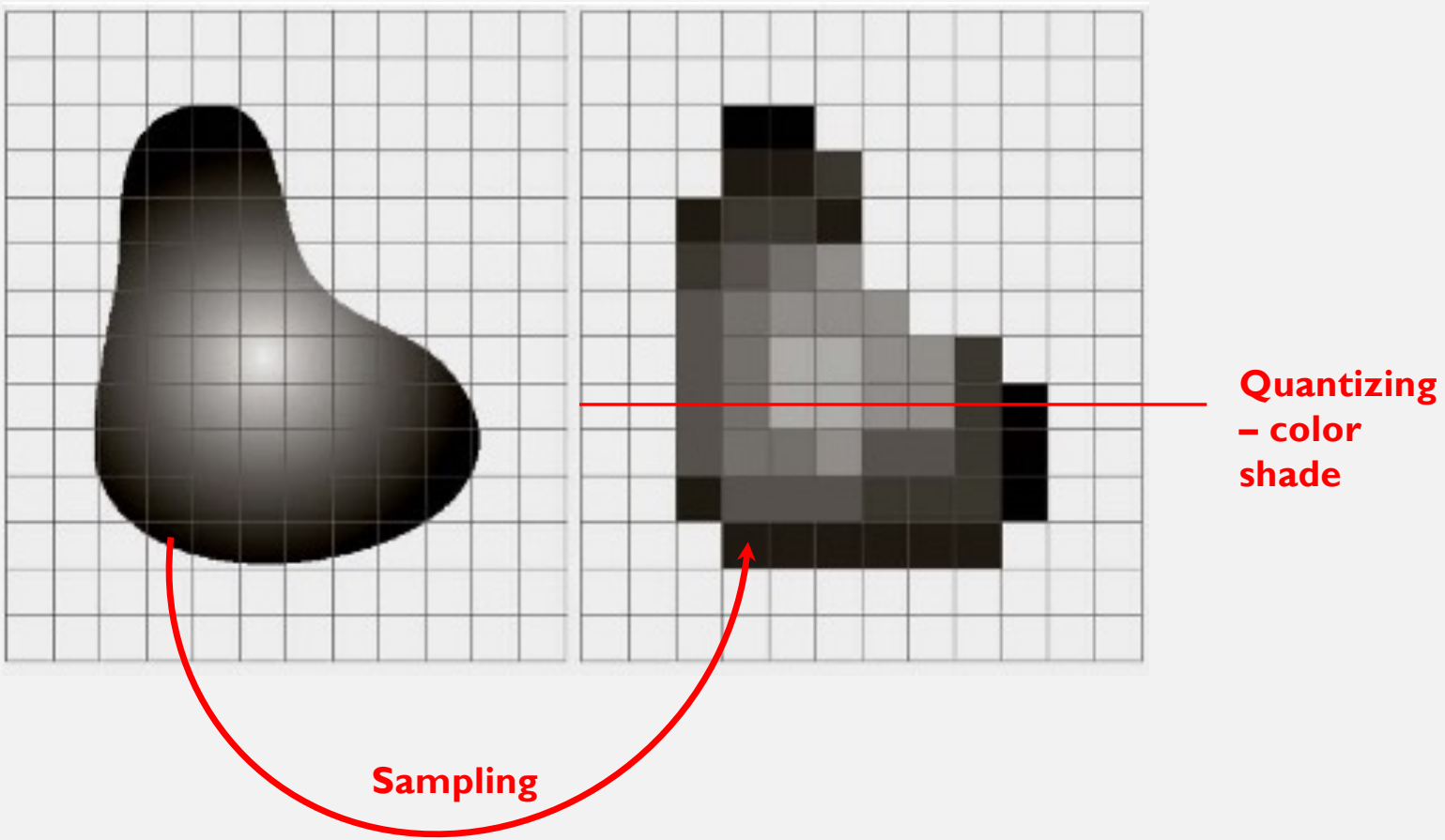


Image from Gonzalez & Woods, Digital Image Processing

IMAGE SAMPLING AND QUANTIZATION



DIGITAL IMAGE

- An image is denoted by two-dimensional function of the form:

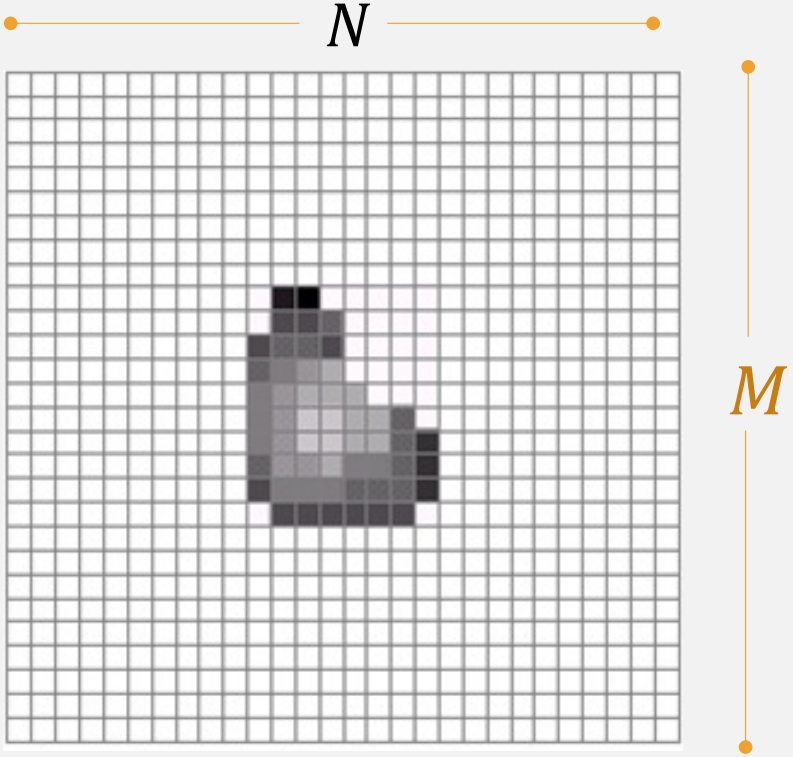
$$f(x, y)$$

- f – amplitude at the spatial dimension (x,y)
- Monochromatic image (grayscale)

$$0 \leq f(x, y) \leq L - 1$$

where L is the total number of intensity levels.

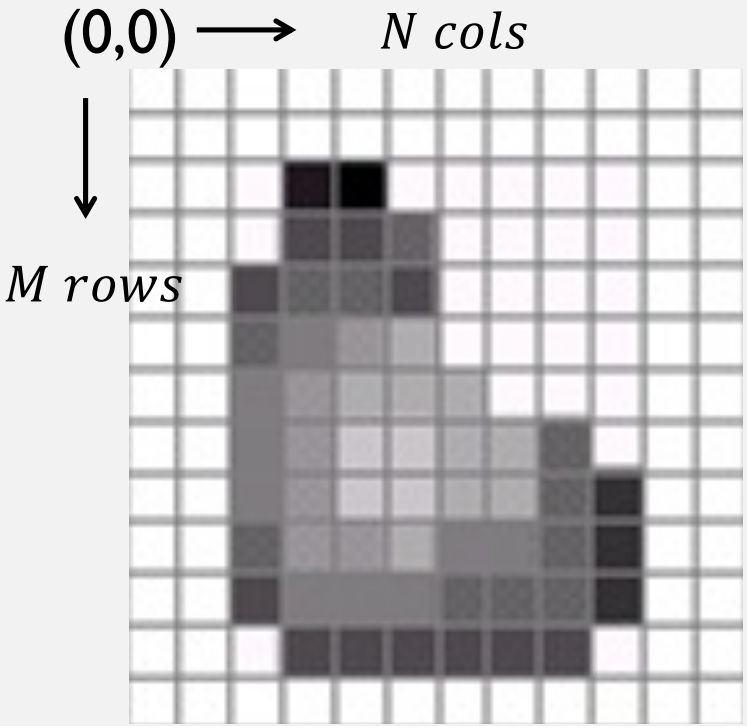
3 bits, L=8



DIGITAL IMAGE REPRESENTATION

- From sampling & quantizing: a matrix of real numbers.
- Digital image **f(x,y)** has **M** rows and **N** columns.
- Image origin (0,0)

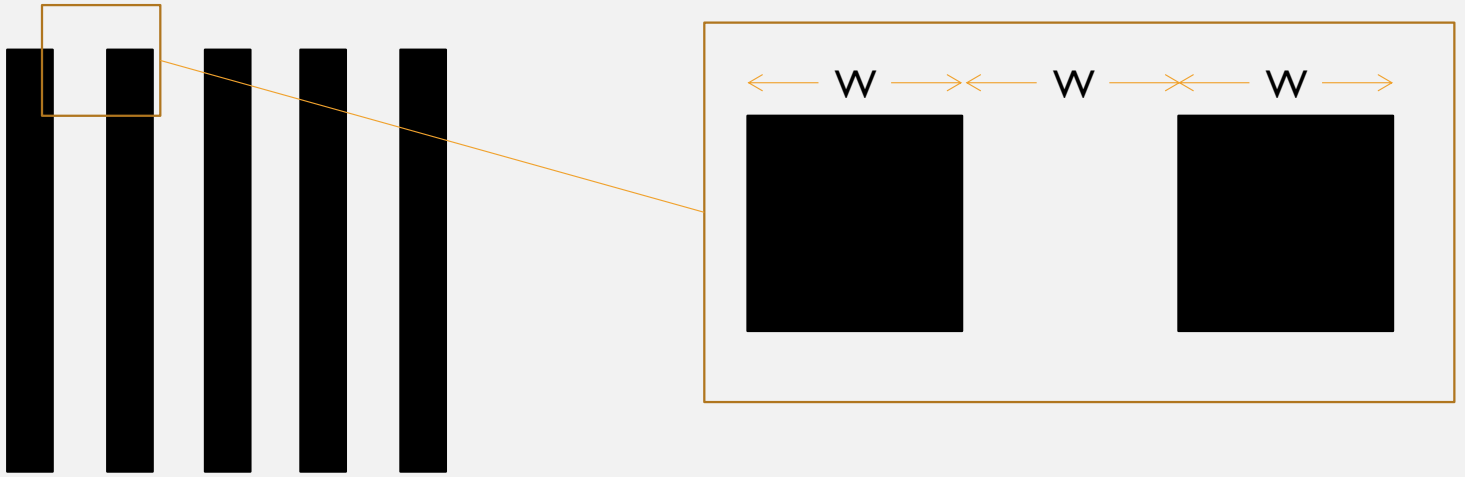
$$f(x,y) = \begin{bmatrix} f(0,0) & \cdots & f(0,N-1) \\ \vdots & \ddots & \vdots \\ f(M-1,0) & \cdots & f(M-1,N-1) \end{bmatrix}$$



how detailed we < sampling / quantize

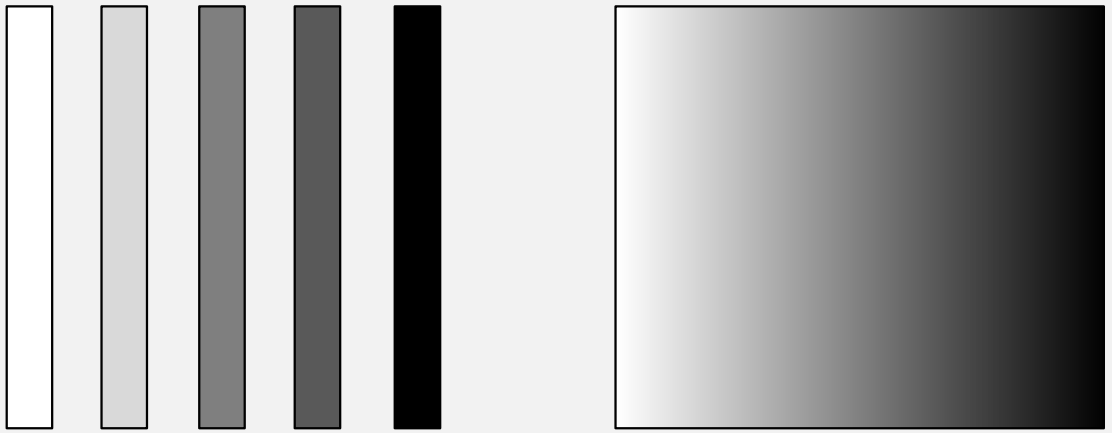
SPATIAL AND GRAY-LEVEL RESOLUTION

- Sampling – principal factor determining the **spatial resolution** of an image.
  - **Spatial resolution** is defined by the smallest discernable details in the image
  - A line pair: width = 2W; 1/2W line pair per unit distance
  - Definition of resolution: smallest num of line pairs per unit distance



SPATIAL AND GRAY-LEVEL RESOLUTION

- **Gray-level** resolution: smallest discernible change in gray level



- Gray-scale: 8 bits (most common) or 16 bits

SPATIAL AND GRAY-LEVEL RESOLUTION

- Spatial Resolution

(by subsampling)

FIGURE 2.19 A 1024 × 1024, 8-bit image subsampled down to size 32 × 32 pixels. The number of allowable gray levels was kept at 256.

(Images from Rafael C. Gonzalez and Richard Wood, Digital Image Processing, 2nd Edition.

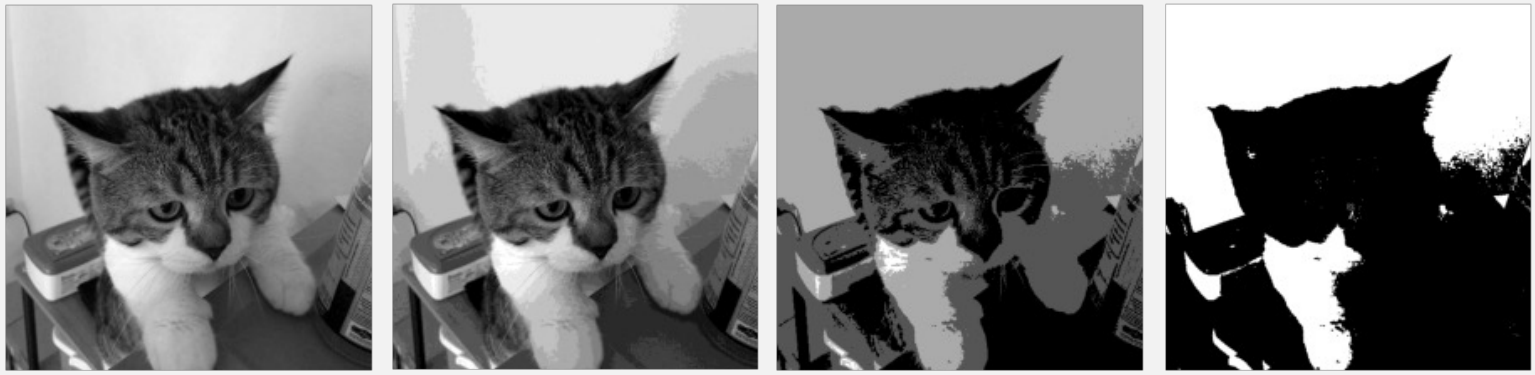
SPATIAL AND GRAY-LEVEL RESOLUTION

- Spatial Resolution

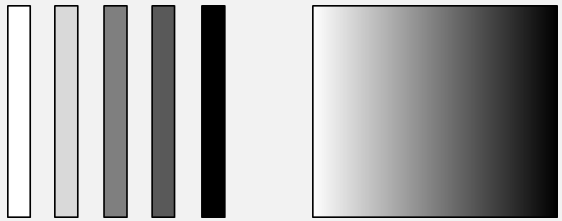
(by subsampling)

SPATIAL AND GRAY-LEVEL RESOLUTION

- Gray-scale resolution:



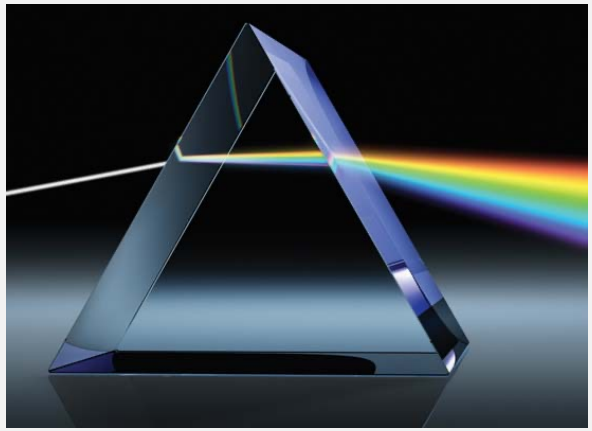
8 bits – 256 levels | 4 bits – 16 levels | 2 bits – 4 levels | 1 bit – 2 levels

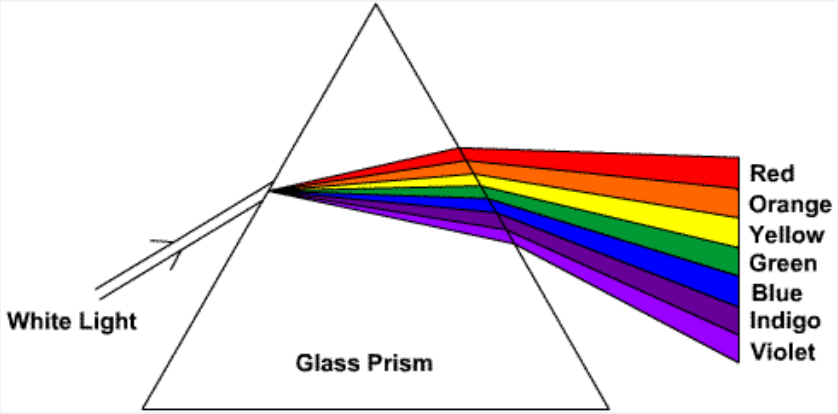


COLOR TRANSFORMATION

COLOR FUNDAMENTALS

- Powerful attribute: simplified object identification from a scene.
- Human can discern thousands of color shades and intensities.



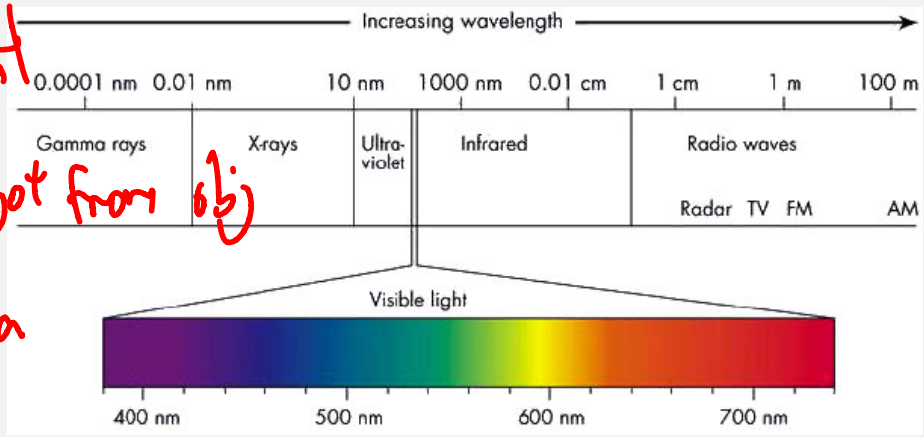


- No color ends abruptly but blends smoothly to the next

Image from Gonzalez & Woods, Digital Image Processing

COLOR FUNDAMENTALS

- If **light** -> **achromatic** (void of color), its attribute is **intensity**.
- **Grayscale** – scalar measure of intensity ranging from black -> gray -> white
- **Chromatic light Source** : 400-700 nm – radiance, luminance and brightness.
  - Radiance – source energy
  - Luminance – observed energy
  - Brightness – color sensation



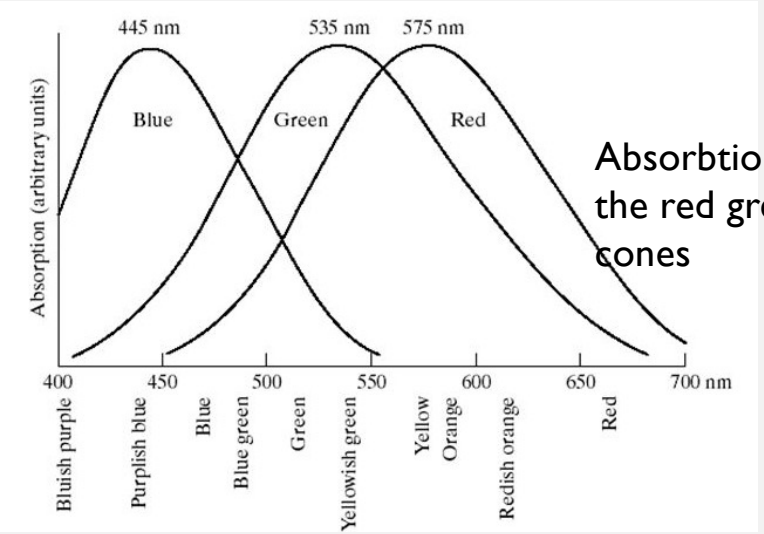
COLOR FUNDAMENTALS

- Cones: responsible for color vision.
  - Three principal sensing categories:
    - 65% red, 33% green and 2% blue (but blue is the most sensitive.
    - Primary color: RGB
    - CIE standard
      - Blue: 435.8 nm
      - Green: 546.1 nm
      - Red: 700 nm

      Colors are seen as variable combination

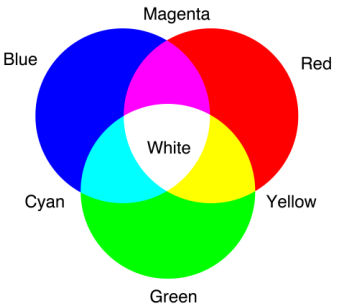
      Of the primary colors (RGB)

standard wave length



Absorbtion of light by the red green and blue cones

Image from Gonzalez & Woods, Digital Image Processing



COLOR FUNDAMENTALS

One color = brightness + chromaticity

- **Primary colors**: added to produce the **secondary colors** : magenta, cyan, and yellow.
- Characteristics distinguish one color from another:
  - **Brightness**: intensity. achromatic component
  - **Hue**: dominant wavelength in a mixture of light waves. dominant color as perceived by an observer, e.g., red, orange, or yellow.
  - **Saturation**: relative purity/amount of white light mixed with the a hue.
  - Hue and saturation: chromaticity

COLOR FUNDAMENTALS

- Amount of red, green and blue needed to form a particular color are called **tristimulus** values denoted as X, Y, and Z, respectively.
- Tristimulus coefficients:

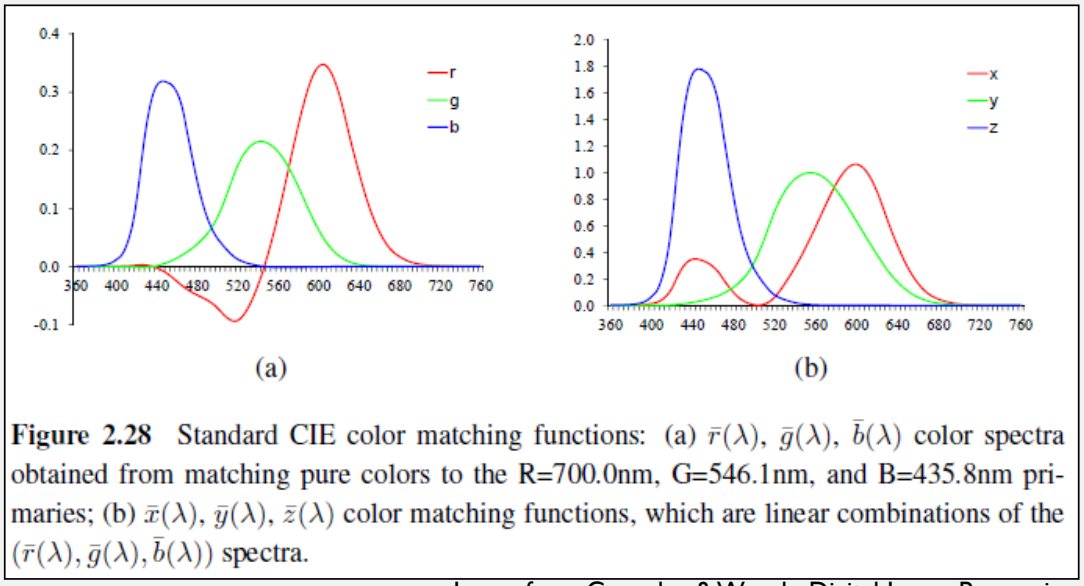


Figure 2.28 Standard CIE color matching functions: (a) $\bar{r}(\lambda)$, $\bar{g}(\lambda)$, $\bar{b}(\lambda)$ color spectra obtained from matching pure colors to the R=700.0nm, G=546.1nm, and B=435.8nm primaries; (b) $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, $\bar{z}(\lambda)$ color matching functions, which are linear combinations of the $(\bar{r}(\lambda), \bar{g}(\lambda), \bar{b}(\lambda))$ spectra.

Image from Gonzalez & Woods, Digital Image Processing

$$x = \frac{X}{X+Y+Z} \qquad y = \frac{Y}{X+Y+Z} \qquad z = \frac{Z}{X+Y+Z}$$

$$x + y + z = 1$$

COLOR FUNDAMENTALS

- CIE chromaticity diagram used to specify colors.
  - Color composition as a function of x (red), y (green)
  - z = 1-(x+y)
  - Green + → 62% green, 25% red and 13% blue
- The point of equal energy is 'equal fractions' of the three (0.33). Saturation at this point is zero.

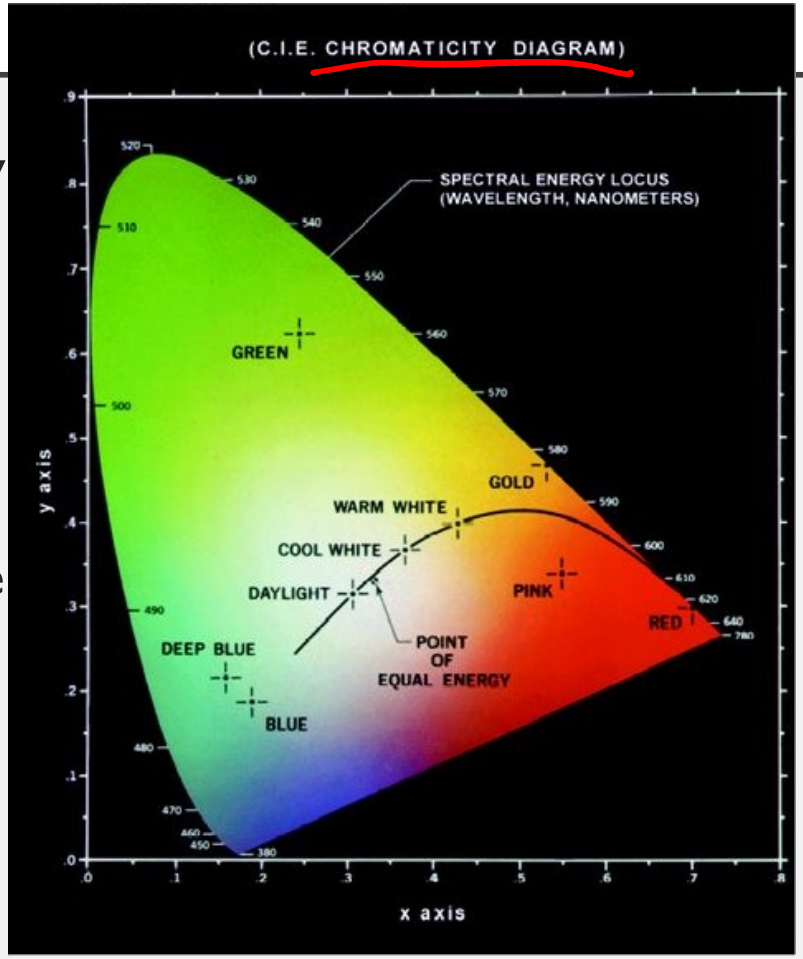


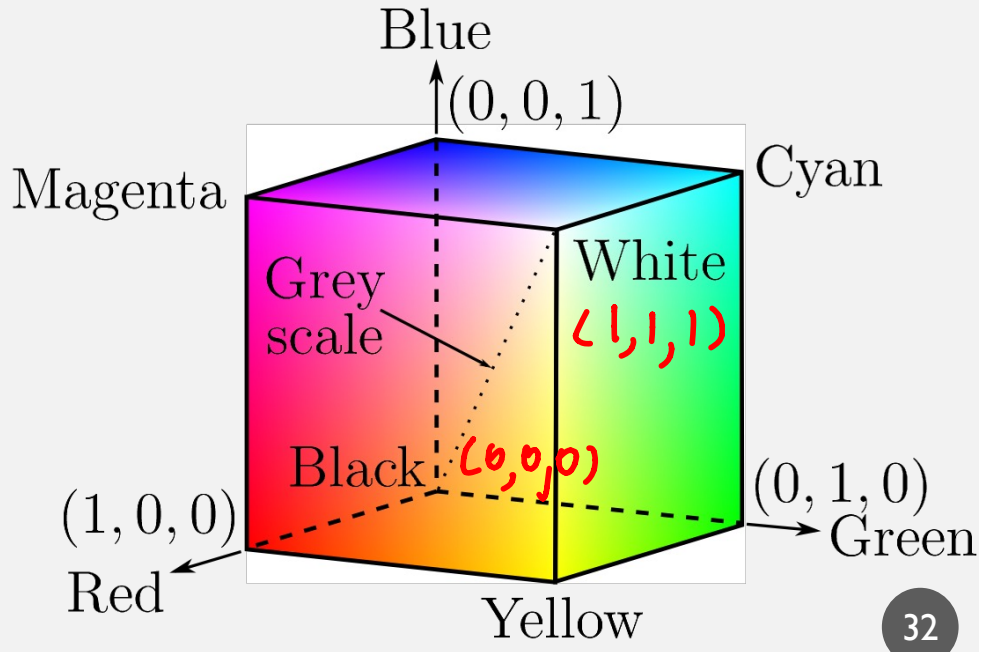
Image from Gonzalez & Woods, Digital Image Processing

COLOR MODELS

- Color space/color system
- RGB – color monitors, broad class of color video cameras
- CMYK – models for color printing
- HSI/HSV – close to how human interpret colors which decouples gray and color components.

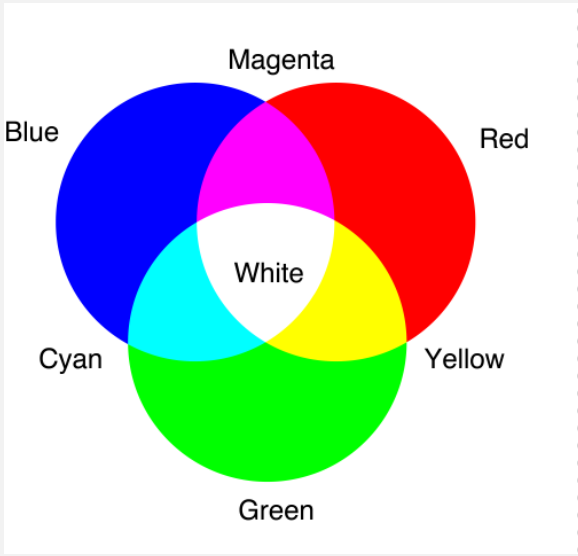
COLOR MODELS

- RGB color model
  - All color values are normalised in a range [0 1].
  - Number of bits – pixel depth
  - 8-bit images = 24-bit depth
  - Full color image: 24 bit
    - $(2^8)^3$ = 16,777,216 colors

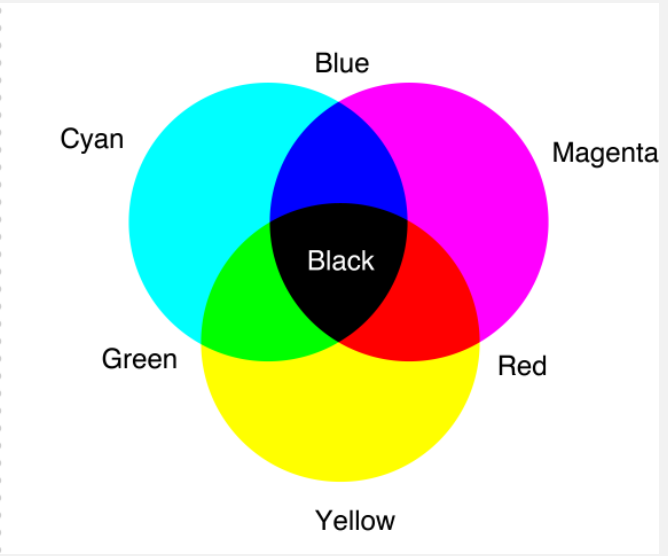


COLOR MODELS

- CMY and CMYK
  - Secondary colors of light (primary colors of pigments)



Mixture of light



Mixture of pigments

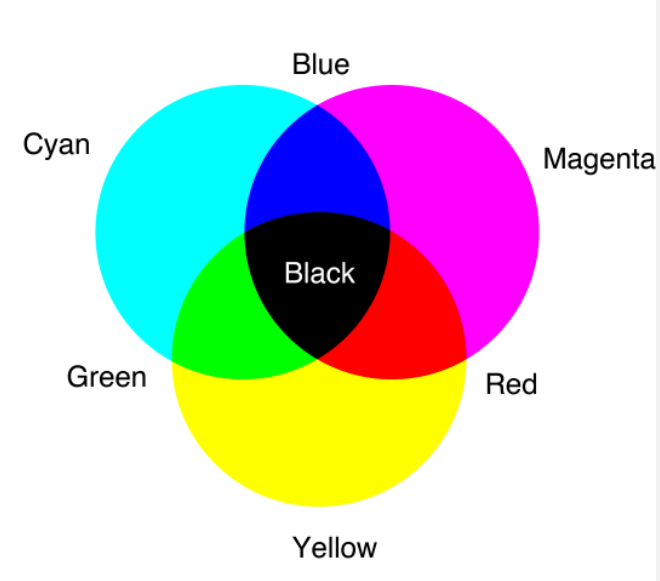
COLOR MODELS

- CMY and CMYK
  - Secondary colors of light (primary colors of pigments)

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

  - K – black – equal amount of CMY for printing

ใช้ black ink โดยตรงแทน: เพราะ: CMY ผสมกัน ≠ black



Mixture of pigments

COLOR MODELS

- RGB & CMY for hardware implementation
- HSI/HSV - better describing colors practically for human interpretation.
- An object – hue, saturation and brightness
- H • Hue: pure color
- S • Saturation: degree hue diluted by white light
- I • Brightness/values: intensity.. Most useful for monochromatic images
- HSI/HSV – ideal tool for developing image processing algorithms..

COLOR MODELS

- HSI/HSV – Hue, Saturation, Intensity/Value

$$H = \begin{cases} \theta & if\ B \le G \\ 360 - \theta & if\ B > G \end{cases}$$

$$\theta = cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{1/2}}\right\}$$

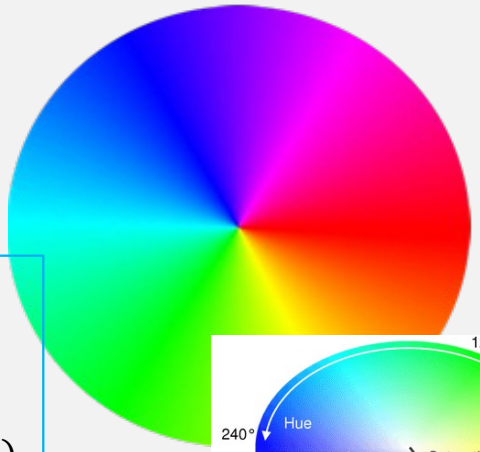
saturation

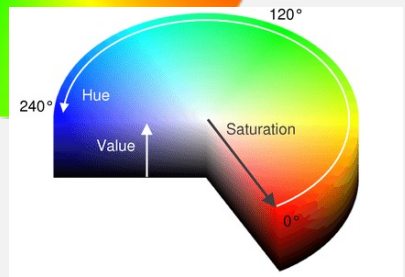
$$S_{HSV} = \begin{cases} 0, & if\ C = 0 \\ \frac{C}{V}, & Otherwise \end{cases}$$

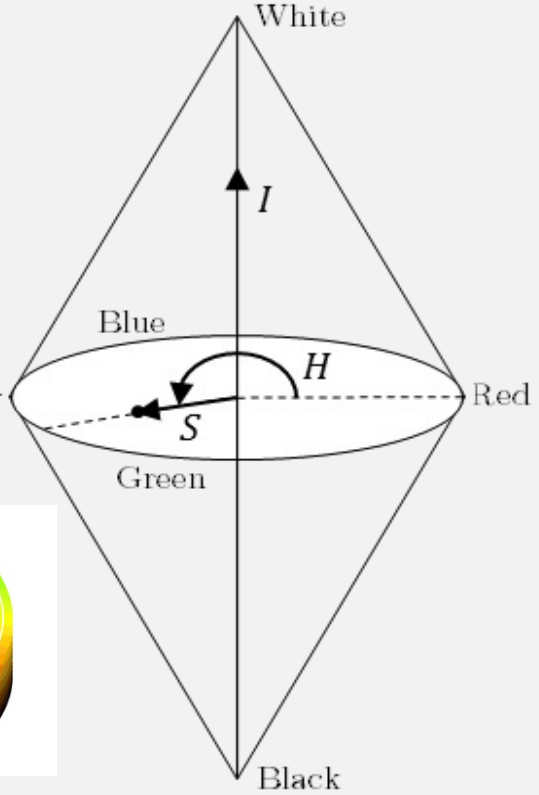
$$C = \max(R,G,B) - \min(R,G,B)$$

$$S_{HSI} = 1 - \frac{3 * min(R,G,B)}{I}$$

$$I = \frac{1}{3}(R+G+B) \qquad V = max\ (R,G,B)$$







COLOR MODELS

- Gray-level
  - The effective representations in describing a monochromatic image

30% 60% 11%

$$Gray = 0.2989 * R + 0.587 * G + 0.1140 * B$$

COLOR MODELS

- L*a*b* or CIELAB model
  - Device-independent colors
  - Similar to HSV model, L*a*b* colour consists of two main components - lightness (L*) and chromaticity (a* and b*)
    - a* - relative to green-red opponents
    - b* - relative to blue-yellow opponents

ARITHMETIC OPERATIONS

SPATIAL DOMAIN

- Spatial - the aggregate of pixels composing an image.
- Spatial domain method: operate directly on these pixels denoted by

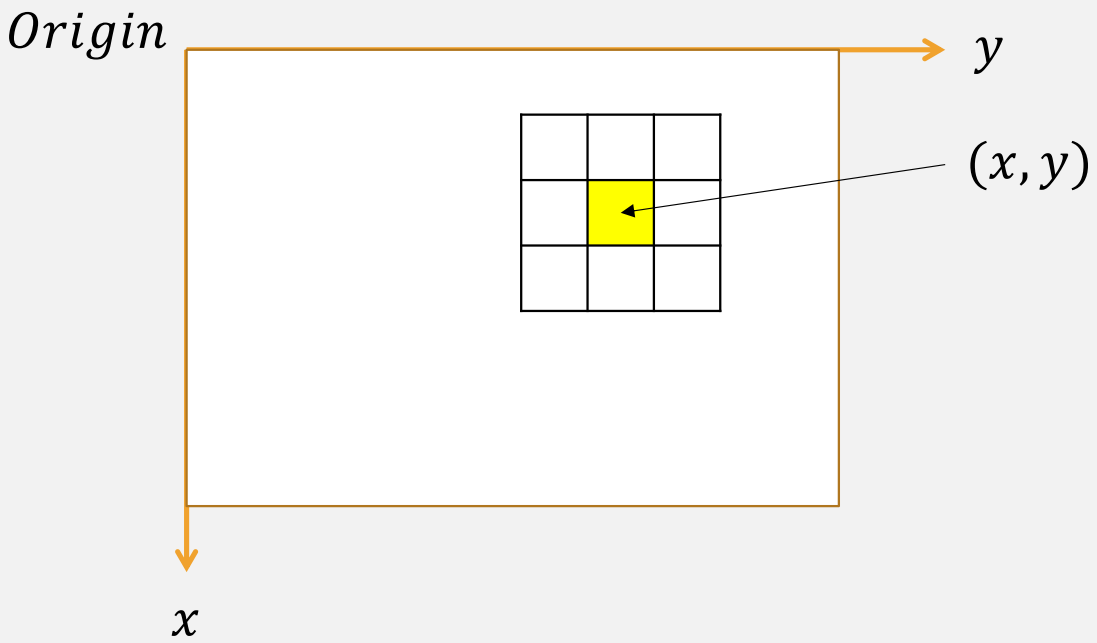
T < (pixel by pixel operation) piece-wise : each pixel
T < neighboring : depend on neighbor pixel

$$g(x,y) = T[f(x,y)]$$

- f(x,y) is the input image, g(x,y) is the processed image, and T is an operator on f defined over some neighborhood of (x,y).

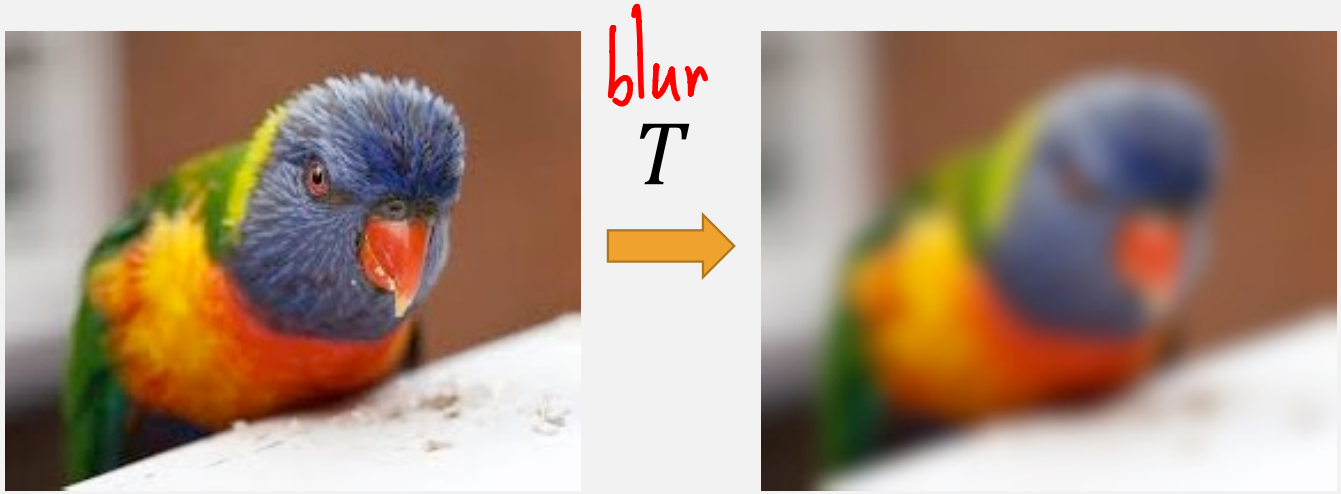
SPATIAL DOMAIN

$$g(x,y) = T[f(x,y)]$$



SPATIAL DOMAIN

$$g(x,y) = T[f(x,y)]$$



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

SPATIAL DOMAIN

- Two commonly used point operators are **multiplication** and **addition**

$$g(x,y) = a\big(f(x,y)\big) + b$$

a: contrast
b: brightness

- a > 0 and b are often called the **gain** and **bias** parameters which can be used to control **contrast and brightness**, respectively.

f(x,y) → (a=2) → g(x,y) = 2 · f(x,y)

0	100
0	100

→

0	200
0	200

↳ contrast ↑

add t100

100	200
100	200

Brightness ↑

SPATIAL DOMAIN

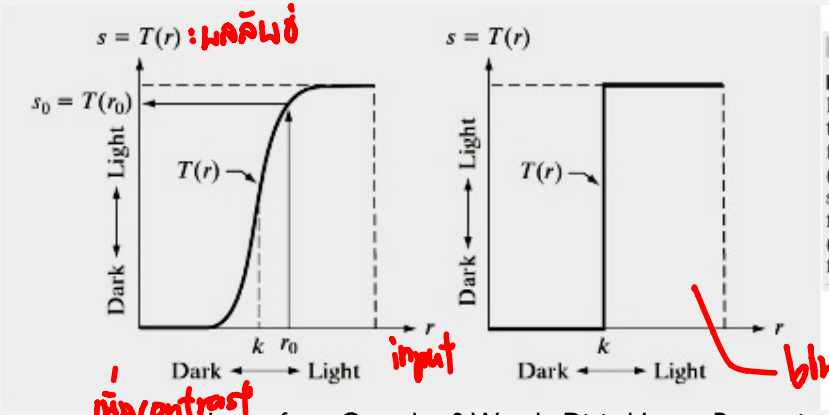
- Gray-level transformation function:

$$s = T(r)$$

- For simplicity in notation, r and s are variables denoting, respectively, the gray level of f(x,y) and g(x,y).
- Example,

  Contrast stretching

  Thresholding function



a b

FIGURE 3.2 Intensity transformation functions. (a) Contrast-stretching function. (b) Thresholding function.

Image from Gonzalez & Woods, Digital Image Processing

BASIC GRAY-LEVEL TRANSFORMATIONS

- Three basic functions used for image enhancement:
  - 1) **Linear**
  - 2) **Logarithmic**
  - 3) **Power-law**

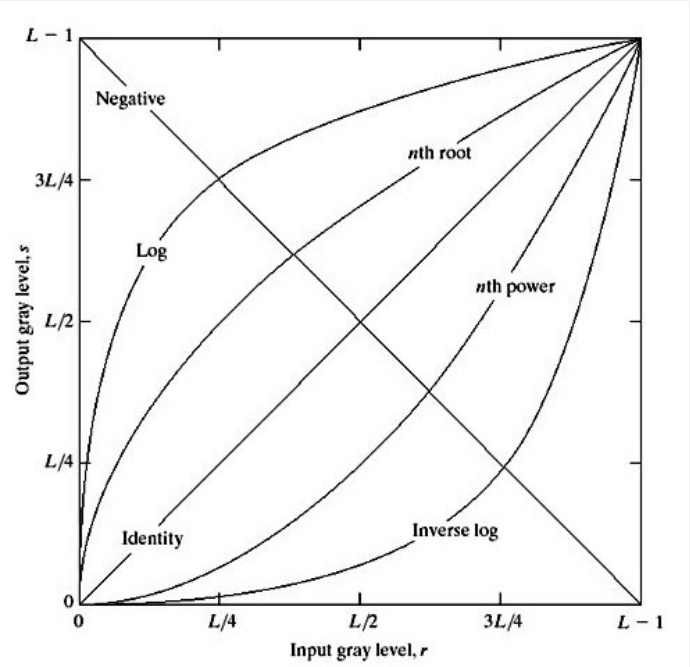
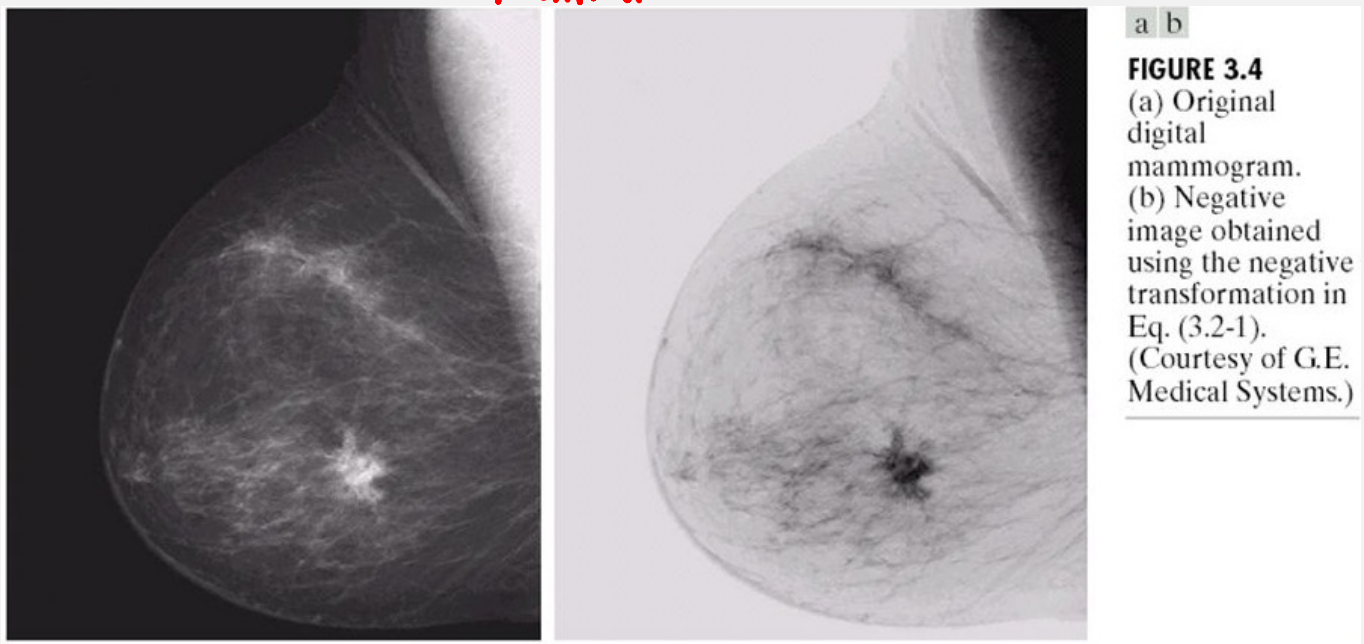


Image from Gonzalez & Woods, Digital Image Processing

BASIC GRAY-LEVEL TRANSFORMATIONS

- **1) Image negatives** – dark area dominant

$$s = (L - 1) - r$$



a b

FIGURE 3.4 (a) Original digital mammogram. (b) Negative image obtained using the negative transformation in Eq. (3.2-1). (Courtesy of G.E. Medical Systems.)

Image from Gonzalez & Woods, Digital Image Processing

BASIC GRAY-LEVEL TRANSFORMATIONS

- **2) Log Transformations**

$$s = c\log(1 + r)$$

c is a constant and assumed r ≥ 0.

- Map narrow range of low gray-level values to a wider range of output levels.

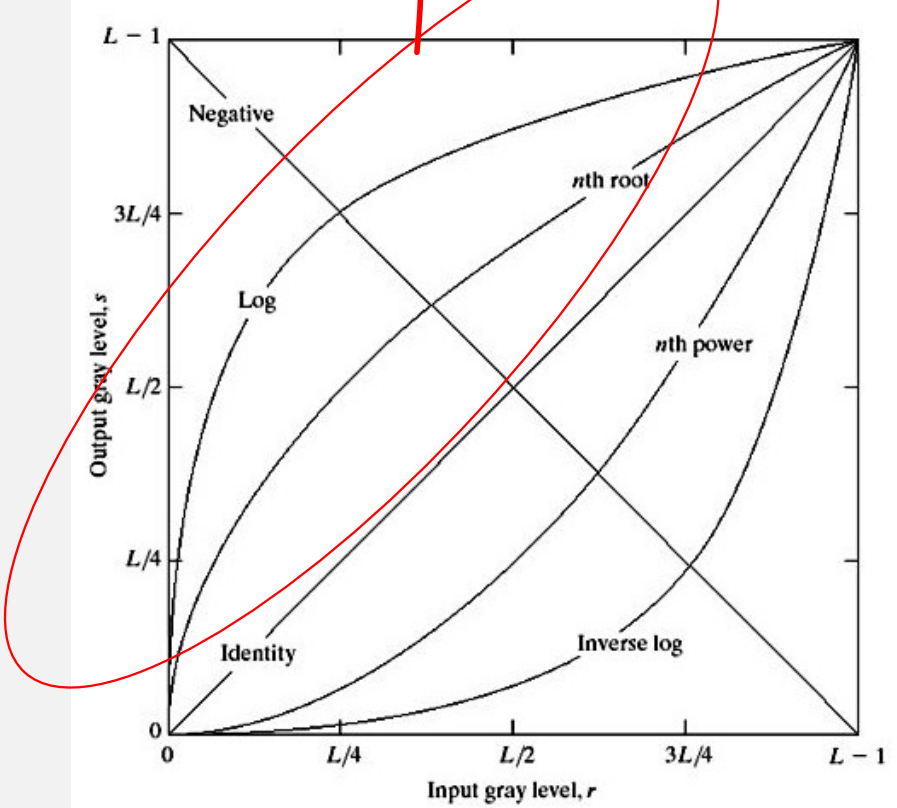


Image from Gonzalez & Woods, Digital Image Processing

BASIC GRAY-LEVEL TRANSFORMATIONS

- **2) Log Transformations**

detailed, brighter

a b

FIGURE 3.5
(a) Fourier spectrum.
(b) Result of applying the log transformation given in Eq. (3.2-2) with $c = 1$.

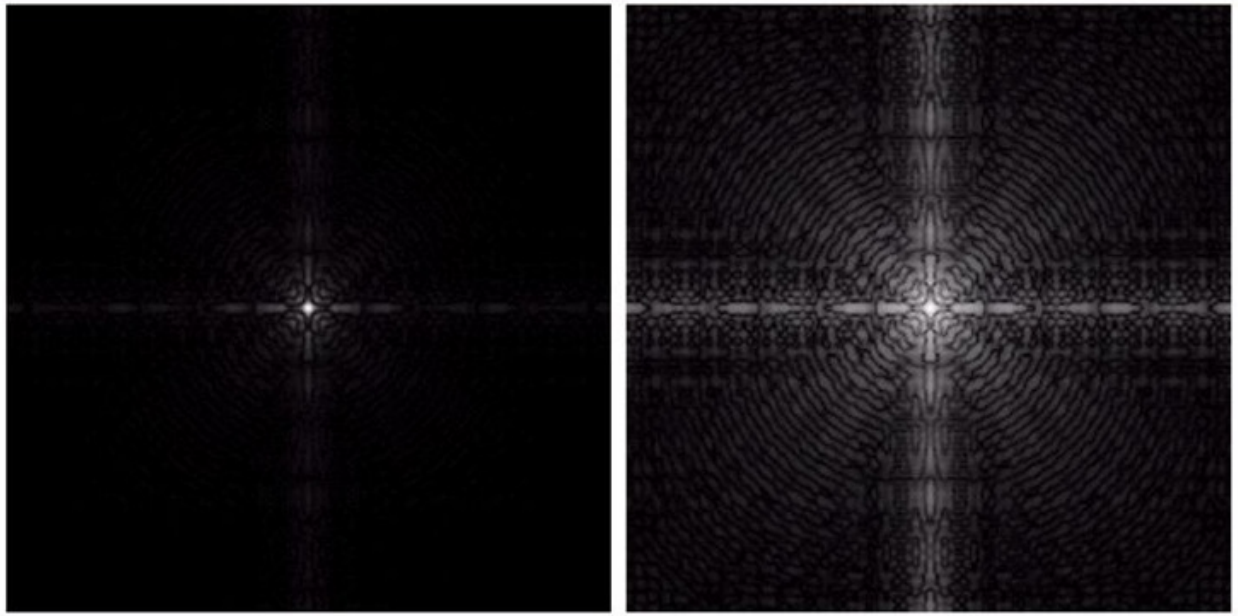


Image from Gonzalez & Woods, Digital Image Processing

BASIC GRAY-LEVEL TRANSFORMATIONS

- **3) Power-Law Transformations**

$$s = cr^{\gamma}$$

- where c and γ are positive constants.
- $\gamma > 1$ opposite to $\gamma < 1$
- $C = \gamma = 1$: identity transformation
- Power-law equation: ***gamma***
- ***Gamma correction***: used in a variety of device.

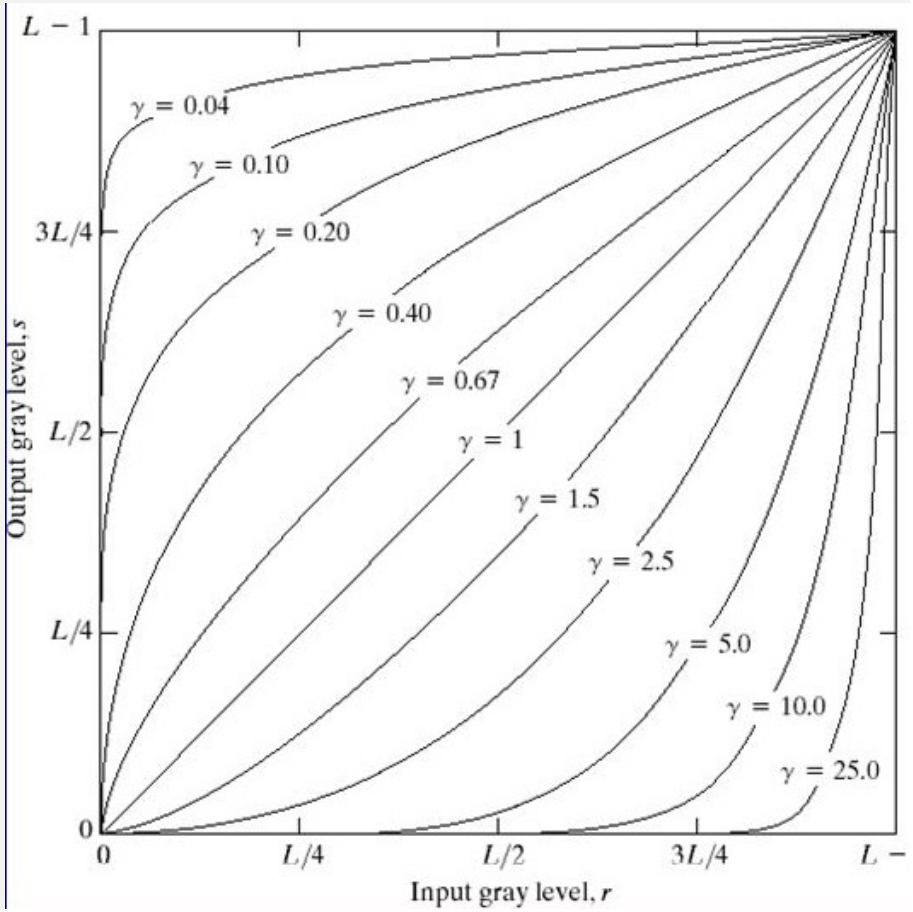
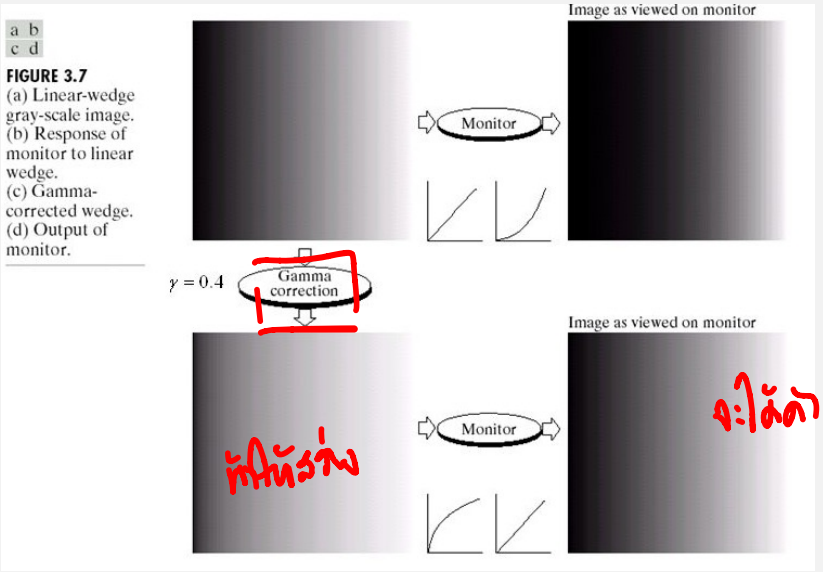


Image from Gonzalez & Woods, Digital Image Processing

BASIC GRAY-LEVEL TRANSFORMATIONS

- **3) Power-Law Transformations**
- The process used to correct this power-law response phenomena is called ***gamma correction***.



a b
c d
FIGURE 3.7
(a) Linear-wedge gray-scale image. (b) Response of monitor to linear wedge. (c) Gamma-corrected wedge. (d) Output of monitor.

Image from Gonzalez & Woods, Digital Image Processing

- In most computer display systems, images are encoded with a gamma of about 0.45 and decoded with the reciprocal gamma of 2.2.
- Macintosh computers, which encoded with a gamma of 0.55 and decoded with a gamma of 1.8

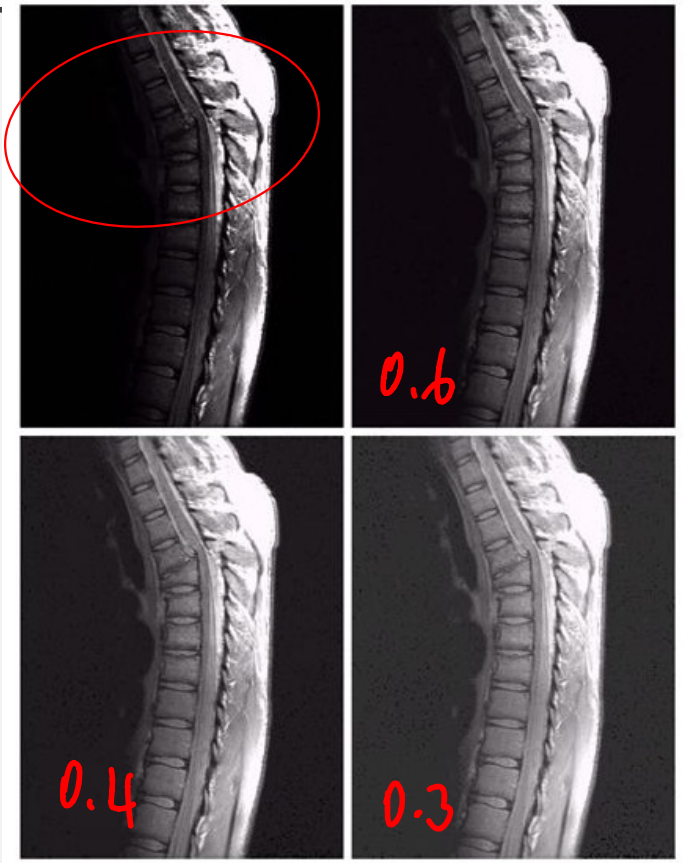
$$s = r^{1/2.5}$$

Gamma correction: display image correctly on a computer screen.

BASIC GRAY-LEVEL TRANSFORMATIONS

brighten some area

- **3) Power-Law Transformations**
- Magnetic resonance image (MRI) of fracture dislocation and spinal cord impingement.
- MRI is predominantly dark; need to expand gray levels.
- γ = 0.4, 0.6 more detail becomes visible



a b
c d

FIGURE 3.8
(a) Magnetic resonance (MR) image of a fractured human spine.
(b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 0.6, 0.4$, and 0.3, respectively. (Original image for this example courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

Image from Gonzalez & Woods, Digital Image Processing

BASIC GRAY-LEVEL TRANSFORMATIONS

- 4) **Piecewise-linear Transformation function**
  - **Contrast Stretching**

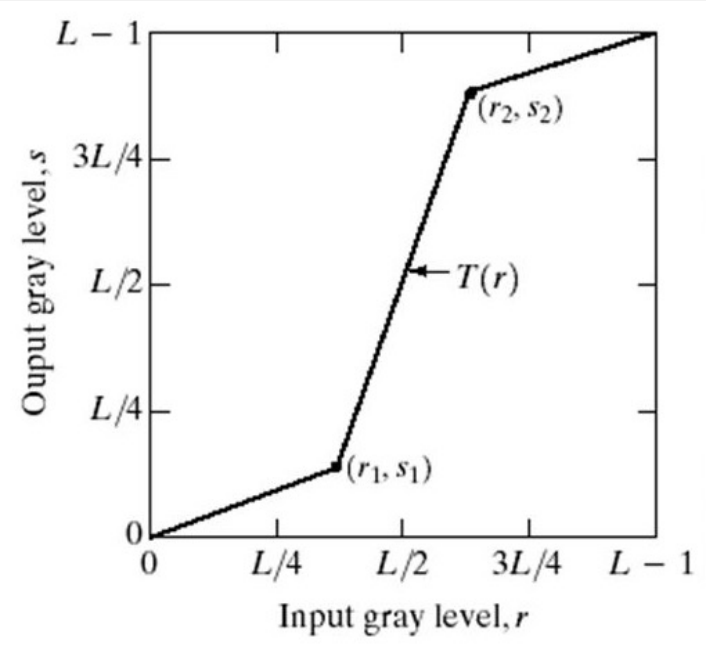
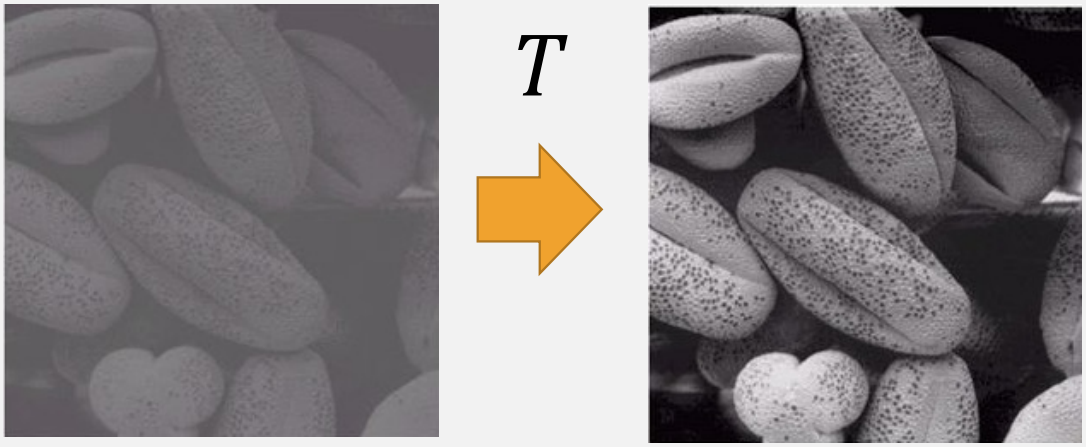


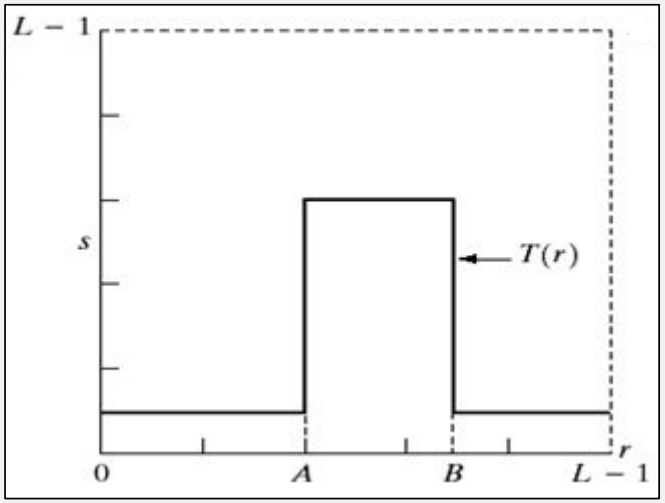
Image from Gonzalez & Woods, Digital Image Processing



BASIC GRAY-LEVEL TRANSFORMATIONS

- **5) Gray-level slicing**
- Highlighting a specific range of gray levels in an image.

เจาะเฉพาะบางส่วนที่ interest



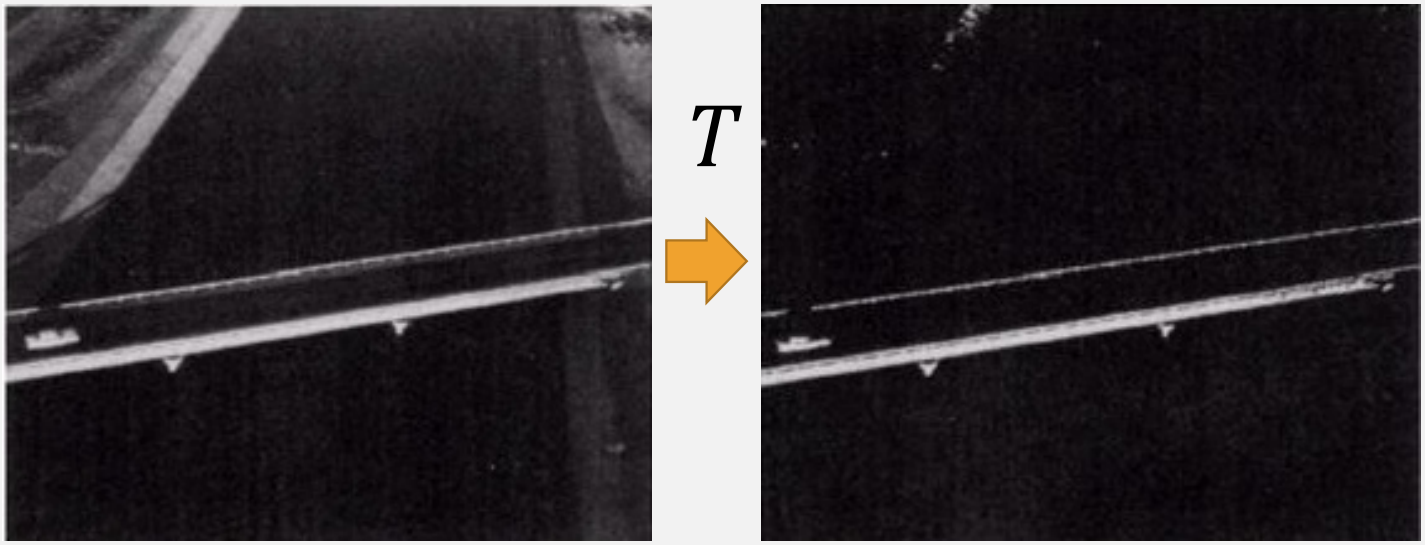


Image from Gonzalez & Woods, Digital Image Processing

BASIC GRAY-LEVEL TRANSFORMATIONS

reduce amount of detailed

- **6) Bit-plane slicing**
  - Plane 0 – least significance – contain more subtle details
  - Plane 7 - most significance – majority of visually significant data
  - Used in image compression

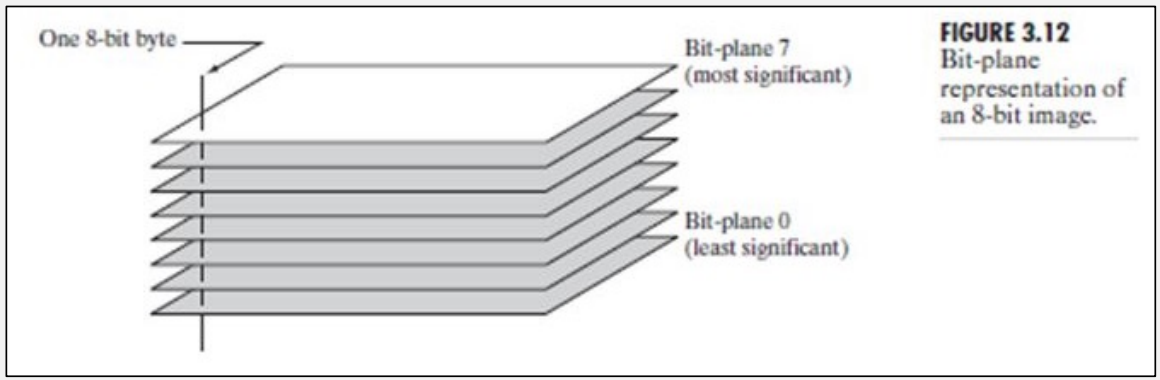


FIGURE 3.12 Bit-plane representation of an 8-bit image.

Image from Gonzalez & Woods, Digital Image Processing

BASIC GRAY-LEVEL TRANSFORMATIONS

- **6) Bit-plane slicing**

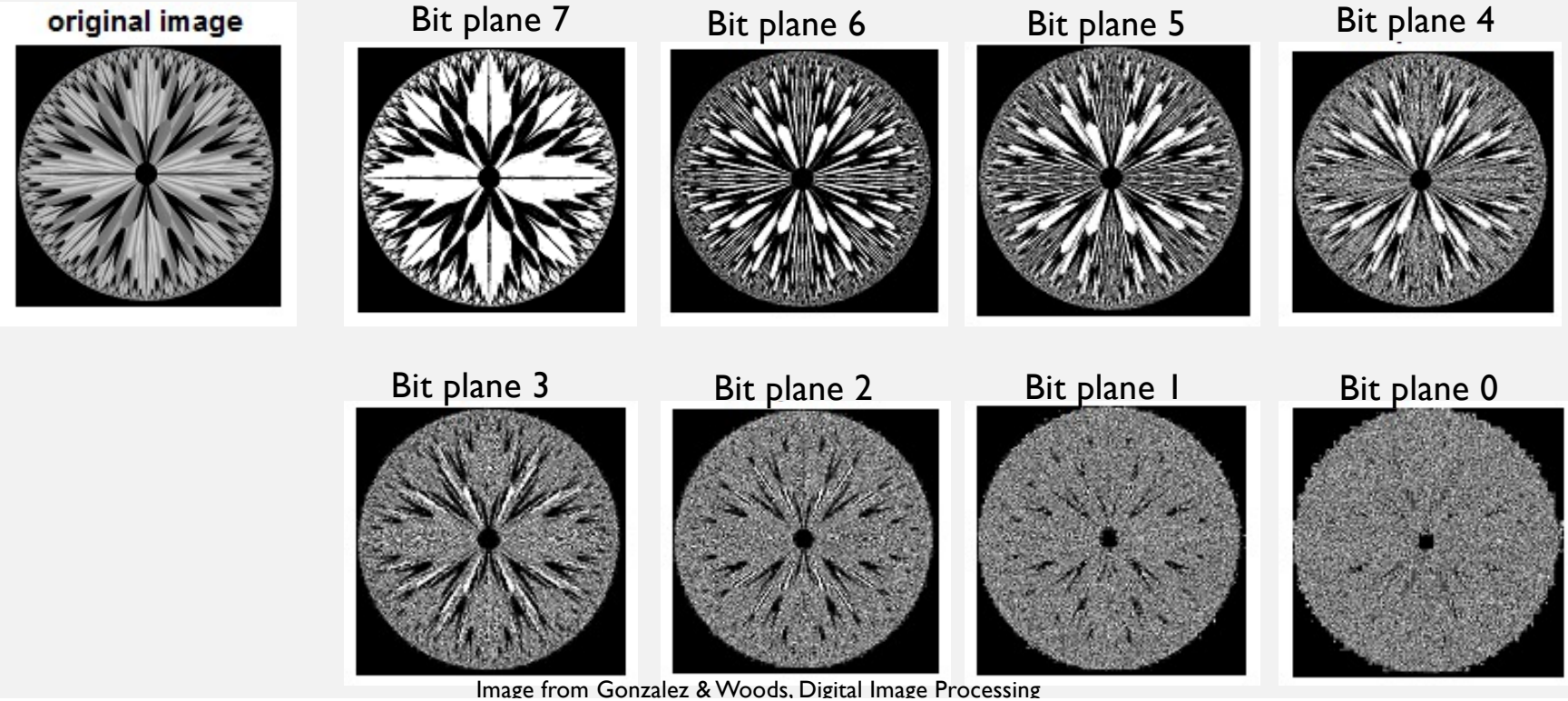
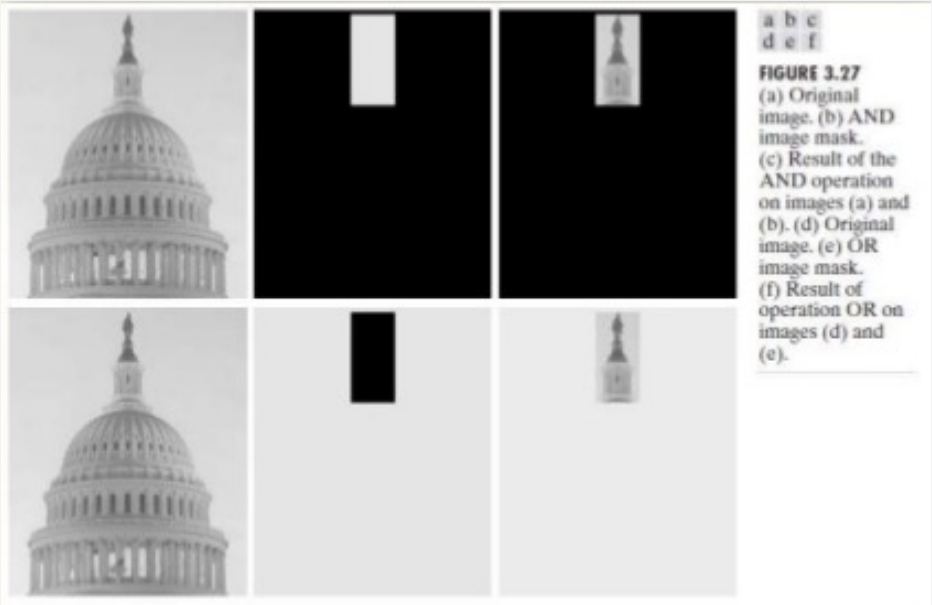


Image from Gonzalez & Woods, Digital Image Processing

ARITHMETIC/LOGIC OPERATIONS

Applications

- Pixel-by-pixel operations between two or more images (except NOT).
- Logic operations: AND, OR used for masking: selecting sub-images in an image.
- Masking: region of interest (ROI)



a b c
d e f

FIGURE 3.27 (a) Original image. (b) AND image mask. (c) Result of the AND operation on images (a) and (b). (d) Original image. (e) OR image mask. (f) Result of operation OR on images (d) and (e).

Image from Gonzalez & Woods, Digital Image Processing

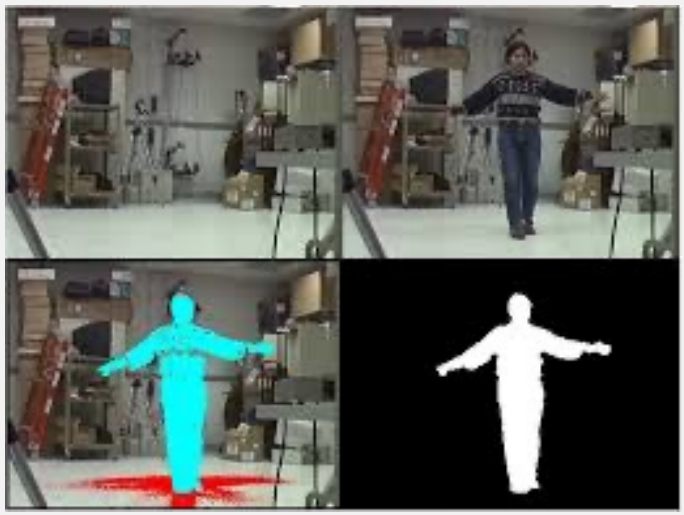
ARITHMETIC/LOGIC OPERATIONS

Applications

- **1) Image Subtraction** –difference between two images f(x,y) and h(x,y):

foreground background

$$g(x,y) = f(x,y) - h(x,y)$$



Horprasert et al., 1999 A Statistical Approach for Real-time Robust Background Subtraction and Shadow Detection

Siricharoen et al., 2010 Vehicle Segmentation using Traffic videos.

Image from Gonzalez & Woods, Digital Image Processing

ARITHMETIC/LOGIC OPERATIONS

Applications

- **2) Image Averaging**
- Considering a noisy image g(x,y) formed by the addition of noise η(x,y) to an original image:

$$g(x,y) = f(x,y) + \eta(x,y)$$

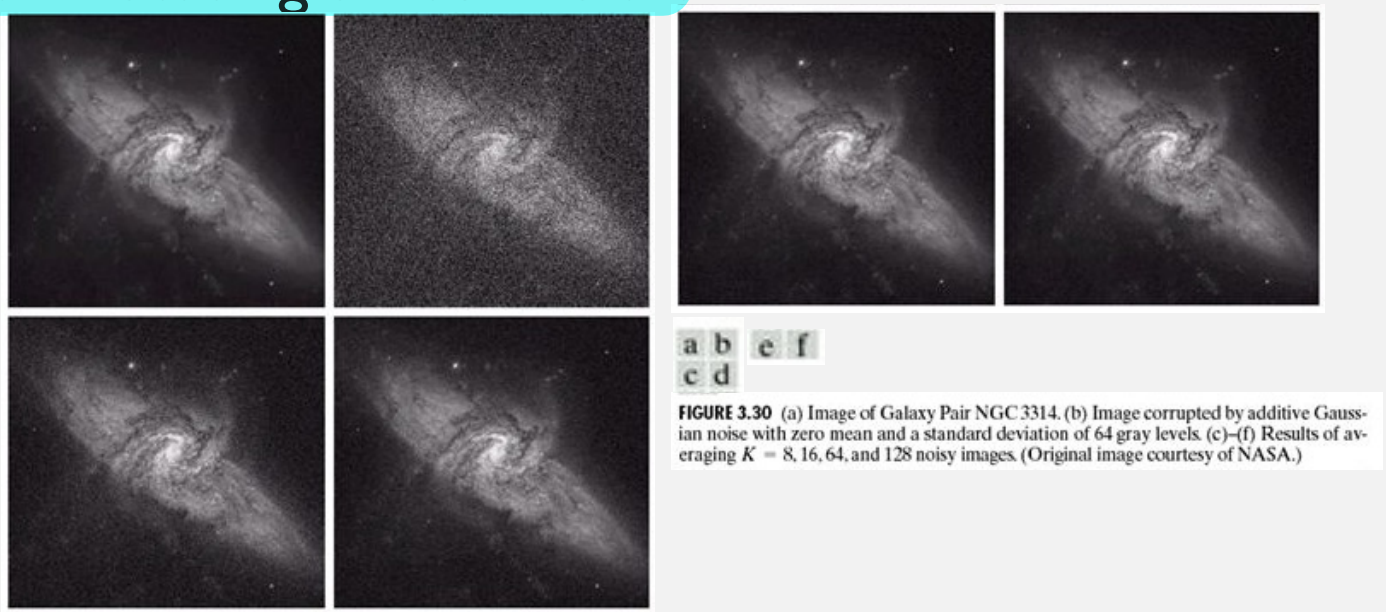
- In this case, noise is uncorrelated and has zero average value.
- One way to reduce the noise, image $\bar{g}$ (x,y) is formed by averaging K different noisy images:

$$\bar{g}(x,y) = \frac{1}{K}\sum_{i=1}^{K} g_i(x,y)$$

ARITHMETIC/LOGIC OPERATIONS

Applications

- **2) Image Averaging** : reduce noise
- **Applications:** Astronomy where imaging with very low light levels is routine causing sensor noise.



a b
c d e f

FIGURE 3.30 (a) Image of Galaxy Pair NGC 3314. (b) Image corrupted by additive Gaussian noise with zero mean and a standard deviation of 64 gray levels. (c)–(f) Results of averaging $K = 8, 16, 64$, and 128 noisy images. (Original image courtesy of NASA.)

Image from Gonzalez & Woods, Digital Image Processing

EXERCISE #1: QUANTIZATION

- **Uniform Quantization**

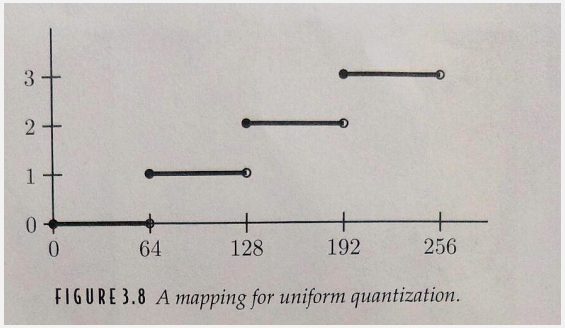
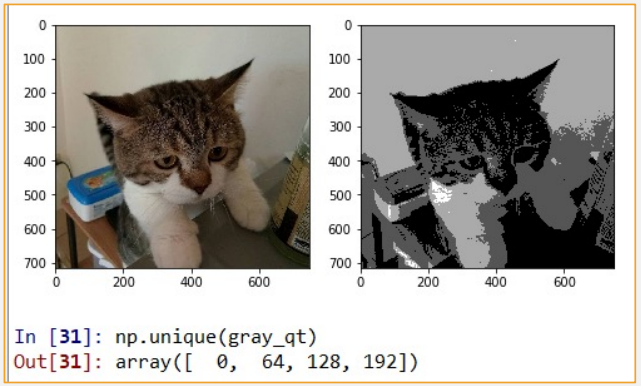


FIGURE 3.8 *A mapping for uniform quantization.*

Original values	Output value
0–63	0
64–127	1
128–191	2
192–255	3

- `img = io.imread("kitty.jpg")`
- `gray = color.rgb2gray(img)`
- `gray255 = np.multiply(gray,255)`
- `gray255[gray255<64] = 0`
- `gray255[(gray255>=64) & (gray255<128)] = 1`
- `gray255[(gray255>=128) & (gray255<192)] = 2`
- `gray255[(gray255>=192)] = 3`



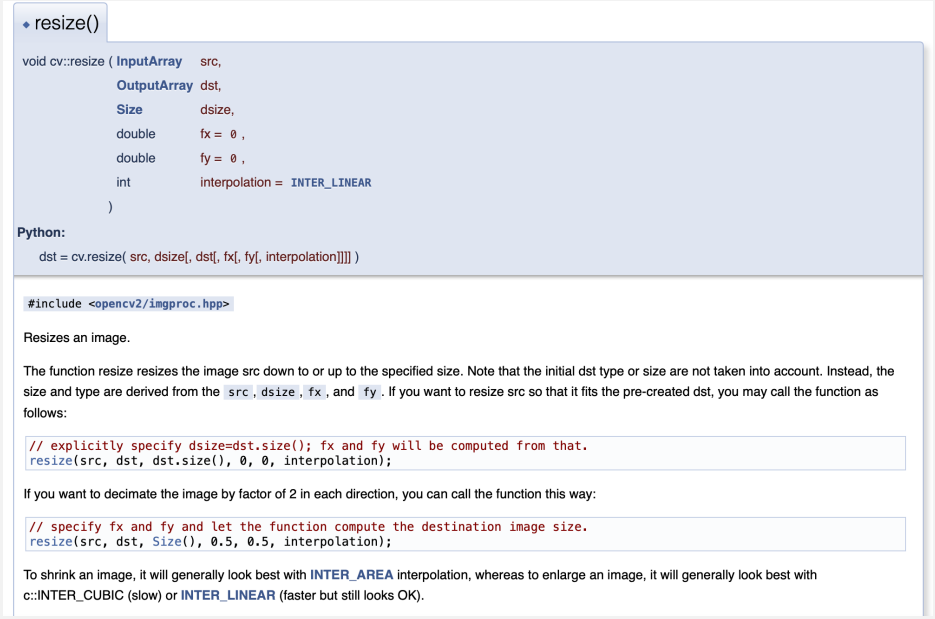
EXERCISE #1: QUANTIZATION

- Use **uniform quantization** to quantize a grayscale image (0-255) to binary image and analyze the results.

EXERCISE #2 SHRINK AND ZOOM IN

- Study and apply `cv2.resize()` to shrink an image down to 20% and input subsampling methods:
  - `nearest` method
  - `bilinear` method
  - `bicubic` method
  - `Area` method

  Compare the results and analyze.



◆ resize()

```
void cv::resize ( InputArray  src,
                  OutputArray dst,
                  Size        dsize,
                  double      fx = 0 ,
                  double      fy = 0 ,
                  int         interpolation = INTER_LINEAR
                )
```

Python:

```
dst = cv.resize( src, dsize[, dst[, fx[, fy[, interpolation]]]] )
```

```
#include <opencv2/imgproc.hpp>
```

Resizes an image.

The function resize resizes the image src down to or up to the specified size. Note that the initial dst type or size are not taken into account. Instead, the size and type are derived from the `src` , `dsize` , `fx` , and `fy` . If you want to resize src so that it fits the pre-created dst, you may call the function as follows:

```
// explicitly specify dsize=dst.size(); fx and fy will be computed from that.
resize(src, dst, dst.size(), 0, 0, interpolation);
```

If you want to decimate the image by factor of 2 in each direction, you can call the function this way:

```
// specify fx and fy and let the function compute the destination image size.
resize(src, dst, Size(), 0.5, 0.5, interpolation);
```

To shrink an image, it will generally look best with INTER_AREA interpolation, whereas to enlarge an image, it will generally look best with c::INTER_CUBIC (slow) or INTER_LINEAR (faster but still looks OK).

Manual:
https://docs.opencv.org/4.5.2/da/d54/group__imgproc__transform.html#ga47a974309e9102f5f08231edc7e7529d

EXERCISE #2 SHRINK AND ZOOM IN

- Shrinking (Resize)
  - img.shape
  - (512, 512, 3)

- Shrink it to 64 x 64 and save to a file comparing nearest neighbor method and area method (naming kitty64.jpg)

- Zoom it to 1000 x 1000 and save to a file comparing nearest neighbor method and bicubic method (naming kitty2048.jpg)

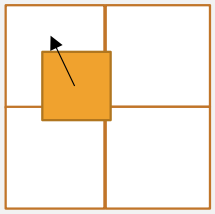
EXERCISE #2 SHRINK AND ZOOM IN

```
SIZE = 64

img_resize_bc = cv2.resize(img,(SIZE, SIZE),interpolation=cv2.INTER_CUBIC)
img_resize_nn = cv2.resize(img,(SIZE, SIZE),interpolation=cv2.INTER_NEAREST)
```

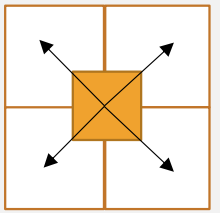
IMAGE INTERPOLATION

- **INTER_NEAREST** - a nearest-neighbor interpolation
- **INTER_LINEAR** - a bilinear interpolation samples all the 4-nearest pixels surrounding the target pixel. The pixel value is calculated as a weighted average of these four pixels (used by default)
- **INTER_CUBIC** - a bicubic interpolation using a polynomial curve fitting on the 4x4 pixel neighborhood
- **INTER_AREA** - resampling using pixel area relation. It may be a preferred method for image decimation, as it gives moire'-free results. But when the image is zoomed, it is similar to the INTER_NEAREST method. (recommended to shrinking an image)

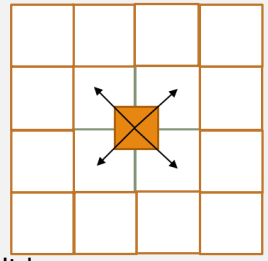
```
[[10 20]]  // Original

// Nearest Neighbouring
[[10 10 10 10 20 20 20]
 [10 10 10 10 20 20 20]
 [10 10 10 10 20 20 20]]
```



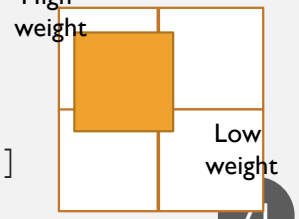
```
// Linear
[[10 10 12 15 18 20 20]
 [10 10 12 15 18 20 20]
 [10 10 12 15 18 20 20]]
```



```
// Cubic
[[ 9 10 12 15 18 20 21]
 [ 9 10 12 15 18 20 21]
 [ 9 10 12 15 18 20 21]]
```

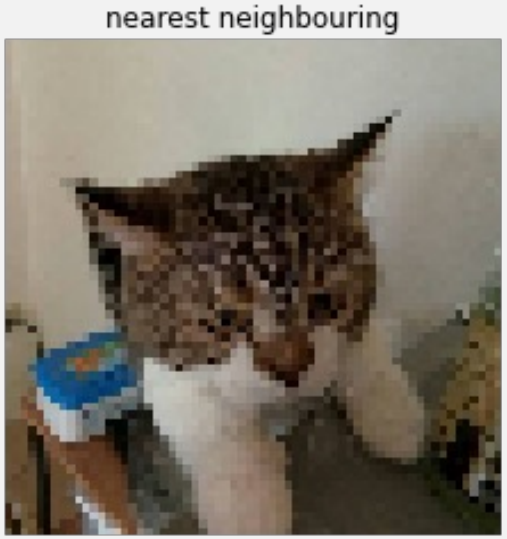


```
// Area
[[10 10 10 15 20 20 20]
 [10 10 10 15 20 20 20]
 [10 10 10 15 20 20 20]]
```

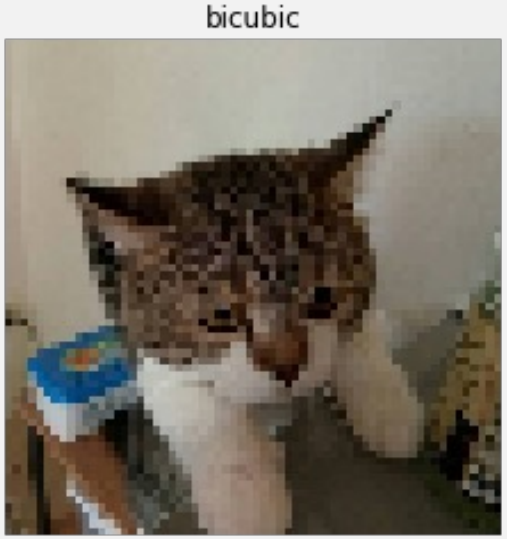


EXERCISE #2 SHRINK AND ZOOM IN

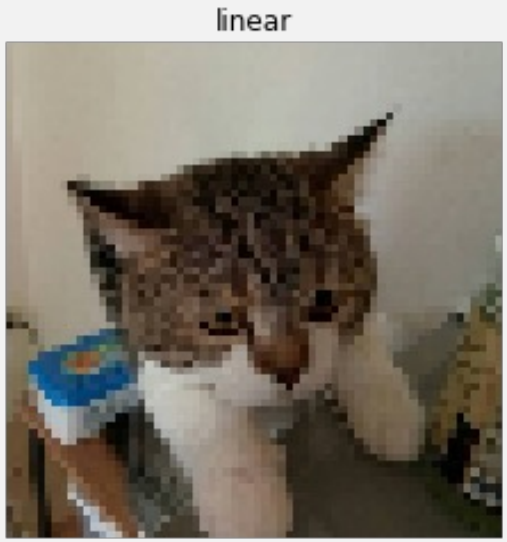
- Shinking an image to 64 x 64



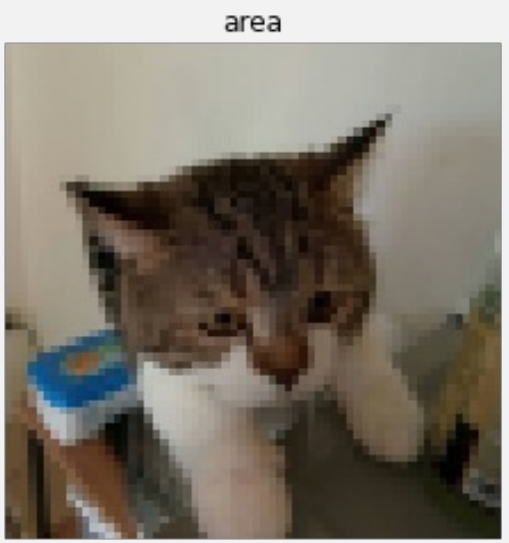
Nearest Neighbouring Interpolation



Bicubic Interpolation



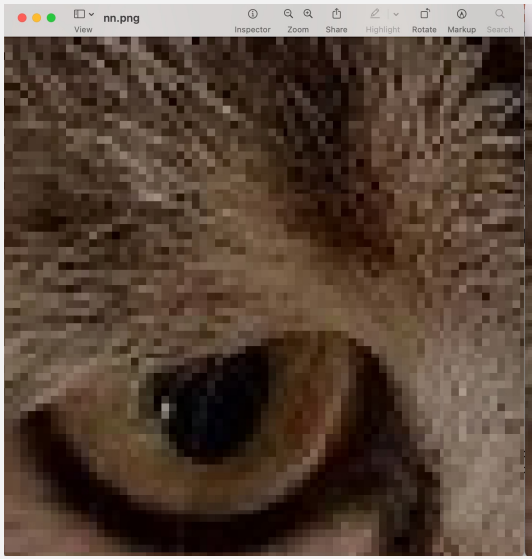
Linear Interpolation



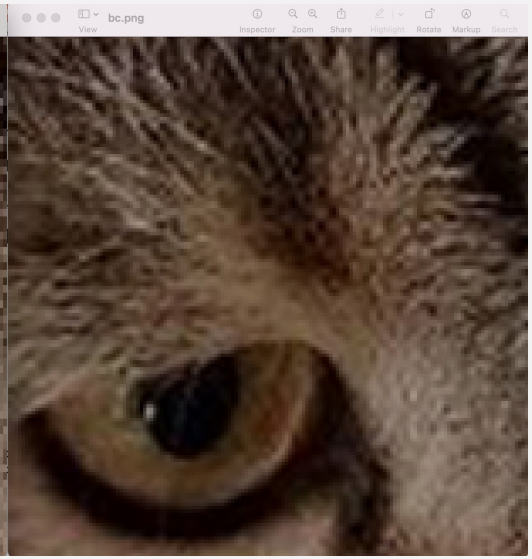
Area Interpolation

EXERCISE #2 SHRINK AND ZOOM IN

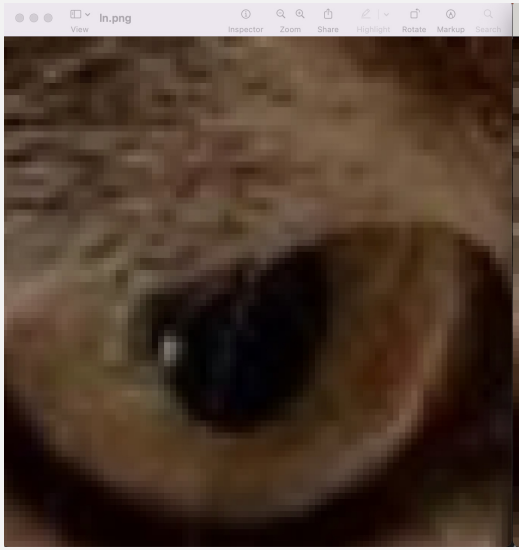
- Zoom in to see more details of an upsampling to 1000 x 1000



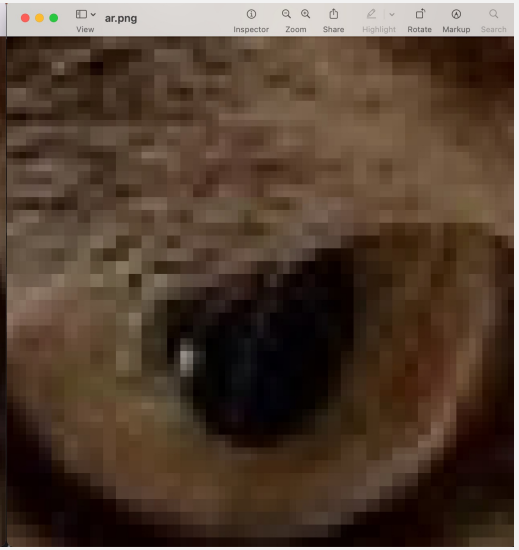
Nearest Neighbouring Interpolation



Bicubic Interpolation



Linear Interpolation



Area Interpolation

EXERCISE #3 ARITHMETIC OPERATIONS

- Two commonly used point operators are **multiplication** and **addition**

$$g(x,y) = a\big(f(x,y)\big) + b$$

- a > 0 and b are often called the **gain** and **bias** parameters which can be used to control **contrast and brightness**, respectively.
- **Gain**:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

GAIN = 2

img = cv2.imread("kitty.jpg")
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

img_div = cv2.divide(img_gray,GAIN)
img_mul = cv2.multiply(img_gray,GAIN)
```

EXERCISE #3 ARITHMETIC OPERATIONS

- **Bias**:

```
BIAS = 50

img_add = cv2.add(img_gray,BIAS)
img_sub = cv2.subtract(img_gray,BIAS)
```

- Use matplotlib to display results and answer the following:
  - What is the difference between gain and bias?
- What will happen when operating with an RGB image?

EXERCISE #3 ARITHMETIC OPERATIONS

https://numpy.org/doc/stable/reference/generated/numpy.multiply.html

numpy.multiply

numpy.multiply(*x1*, *x2*, /, *out=None*, *, *where=True*, *casting='same_kind'*, *order='K'*, *dtype=None*, *subok=True*[, *signature*, *extobj*]) = <*ufunc 'multiply'*> ¶

Multiply arguments element-wise.

Parameters: **x1, x2 : *array_like***

Input arrays to be multiplied. If `x1.shape != x2.shape`, they must be broadcastable to a common shape (which becomes the shape of the output).

out : *ndarray, None, or tuple of ndarray and None, optional*

A location into which the result is stored. If provided, it must have a shape that the inputs broadcast to. If not provided or None, a freshly-allocated array is returned. A tuple (possible only as a keyword argument) must have length equal to the number of outputs.

where : *array_like, optional*

This condition is broadcast over the input. At locations where the condition is True, the *out* array will be set to the ufunc result. Elsewhere, the *out* array will retain its original value. Note that if an uninitialized *out* array is created via the default `out=None`, locations within it where the condition is False will remain uninitialized.

****kwargs**

For other keyword-only arguments, see the ufunc docs.

Returns: **y : *ndarray***

The product of *x1* and *x2*, element-wise. This is a scalar if both *x1* and *x2* are scalars.

https://docs.opencv.org/3.4/d2/de8/group__core__array.html

multiply()

```
void cv::multiply ( InputArray    src1,
                    InputArray    src2,
                    OutputArray   dst,
                    double        scale = 1 ,
                    int           dtype = -1
                  )
```

Python:

cv.multiply(src1, src2[, dst[, scale[, dtype]]]) -> dst

`#include <opencv2/core.hpp>`

Calculates the per-element scaled product of two arrays.

The function multiply calculates the per-element product of two arrays:

$$\texttt{dst}(I) = \texttt{saturate}(\texttt{scale} \cdot \texttt{src1}(I) \cdot \texttt{src2}(I))$$

There is also a **MatrixExpressions** -friendly variant of the first function. See **Mat::mul** .

For a not-per-element matrix product, see gemm .

Note

Saturation is not applied when the output array has the depth CV_32S. You may even get result of an incorrect sign in the case of overflow.

Parameters

src1 first input array.

src2 second input array of the same size and the same type as src1.

dst output array of the same size and type as src1.

scale optional scale factor.

dtype optional depth of the output array

See also

add, **subtract**, **divide**, **scaleAdd**, **addWeighted**, **accumulate**, **accumulateProduct**, **accumulateSquare**, **Mat::convertTo**

EXERCISE #4 COLOR MODELS

- **Chromaticity and Achromaticity**
  - Convert an RGB image to HSV image. Analyse each component in HSV.
  - Convert an RGB image to YCbCr image. Analyse each component in YCbCr.
  - Convert an RGB image to CIELAB image. Analyse each component in CIELAB.

REFERENCES

Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Addison-Wesley

- Chapter 2 – Digital Image Fundamentals
- Chapter 3 – 3.2 Some basic gray level transformations
- Chapter 6 – Color Image Processing (6.2, 6.3 Color models)