Digital Image

Presented by: AIADI Oussama

Outline

- Multimedia data forms
- Why Digital image
- Imaging modalities
- human visual perception
- Electromagnetic Spectrum
- Image Acquisition and formation
- Image Sampling and Quantization

MultiMedia data forms

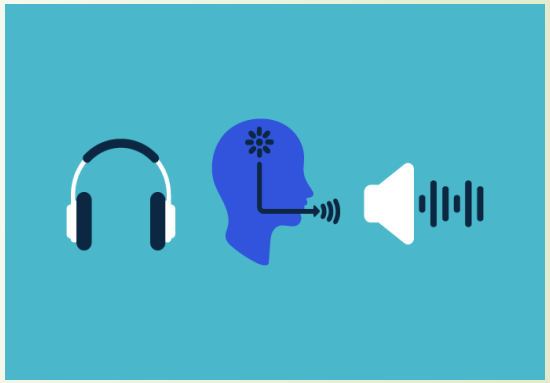
Other than image, there are several data formats, including the following

Image

Video

Image texture, defined as a function of the spatial variation in pixel intensities (gray values), is useful in a variety of applications and has been a subject of intense study by many researchers. One immediate application of image texture is the recognition of image regions using texture properties. For example, in Figure 1(a), we can identify the five different textures and their identities as cotton canvas, straw matting, raffia, herringbone weave, and pressed calf leather. Texture is the most important visual cue in identifying these types of homogeneous regions. This is called *texture classification*. The goal of texture classification then is to produce a classification map of the input image where each uniform textured region is identified with the texture class it belongs to as shown in Figure 1(b). We could also find the texture boundaries even if we could not classify these textured surfaces. This is then the second type of problem that texture analysis research attempts to solve — *texture segmentation*. The goal of texture segmentation is to obtain the boundary map shown in Figure 1(c). *Texture synthesis* is often used for image compression applications. It is also important in computer graphics where the goal is to render object surfaces which are as realistic looking as possible. Figure 2 shows a set of synthetically generated texture images using Markov random field and fractal models [9]. The *shape from texture* problem is one instance of a general class of vision problems known as "shape from X". This was first formally pointed out in the perception literature by Gibson

text



Audio

Why digital image

Digital image has a wide broad of interesting applications, ranging from medical and industrial apps to law enforcement, Hereafter, we shed the light into some of them

Medical diagnosis

Industrial inspection

Control Access

Why digital image

Other applications of digital image include also

Vehicle plate recognition

Historical document restoration

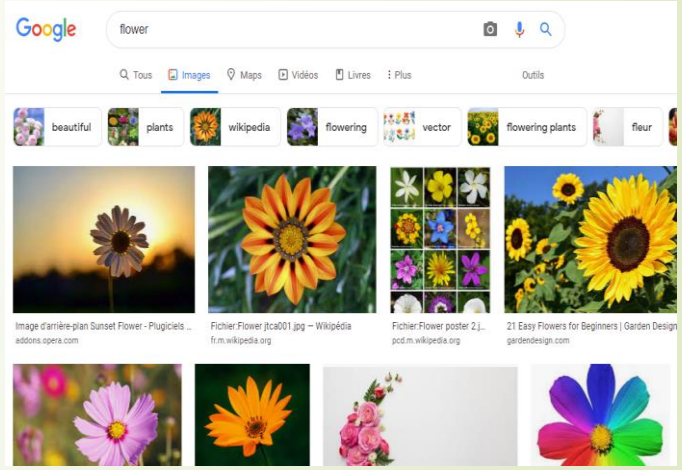


Image retrieval

Why digital image

Looking for more recent apps

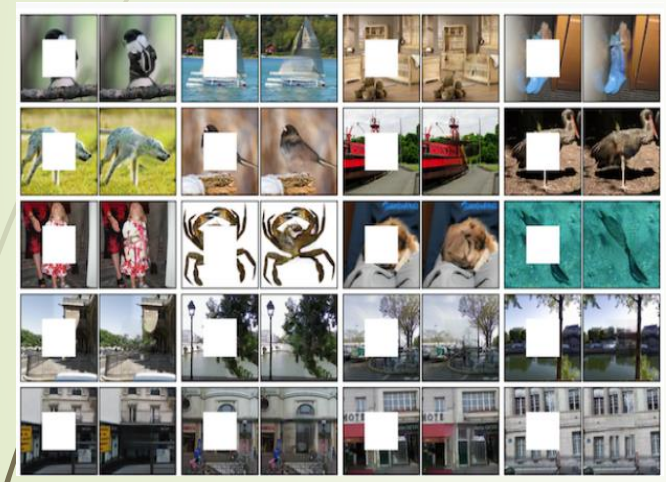


Image inpainting

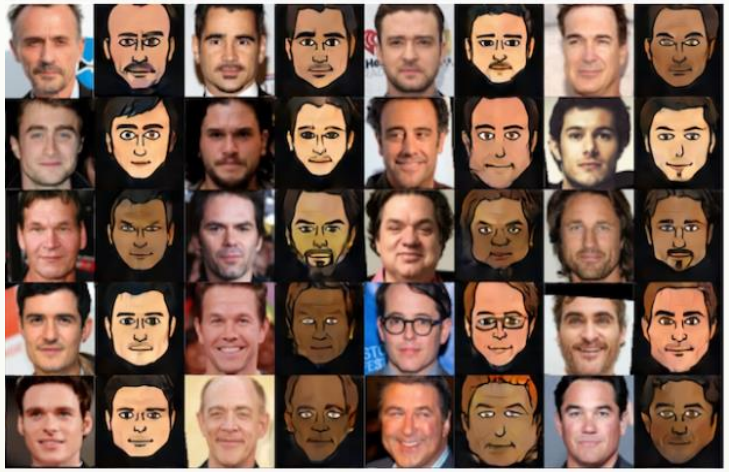
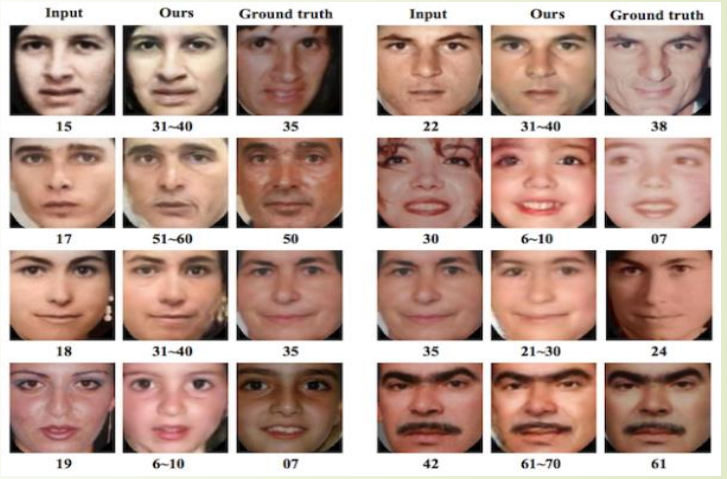


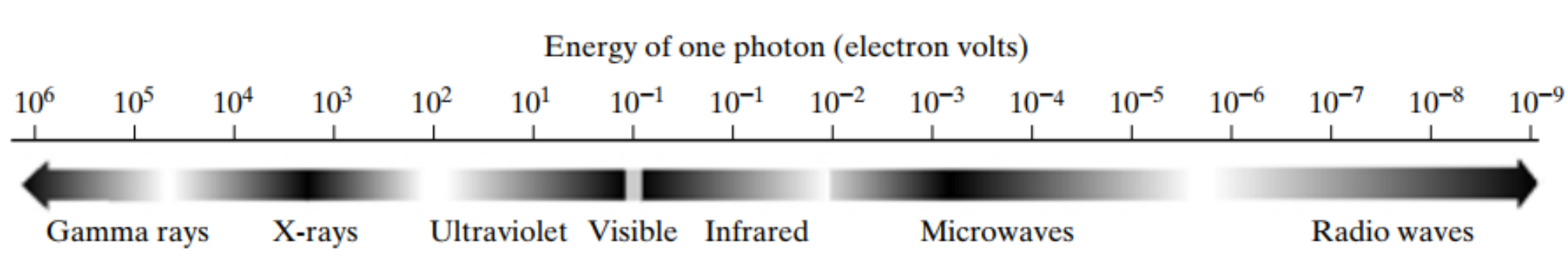
Photo to emojis



Face aging

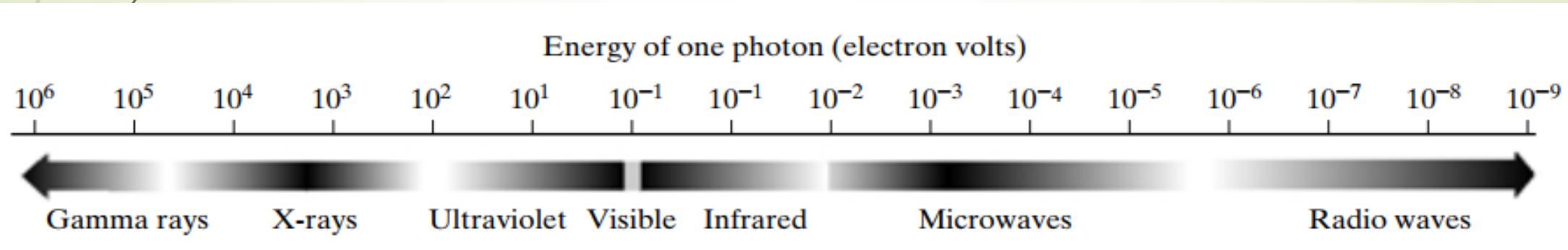
Imaging modalities

Images we see in our daily life can be produced based on different imaging techniques, which differ from each other in terms of energy source. The principal energy source for images in use today is the electromagnetic energy spectrum. Other important sources of energy include acoustic, ultrasonic, and electronic. In addition, synthetic images, used for modeling and visualization, are generated by computer.



Imaging modalities

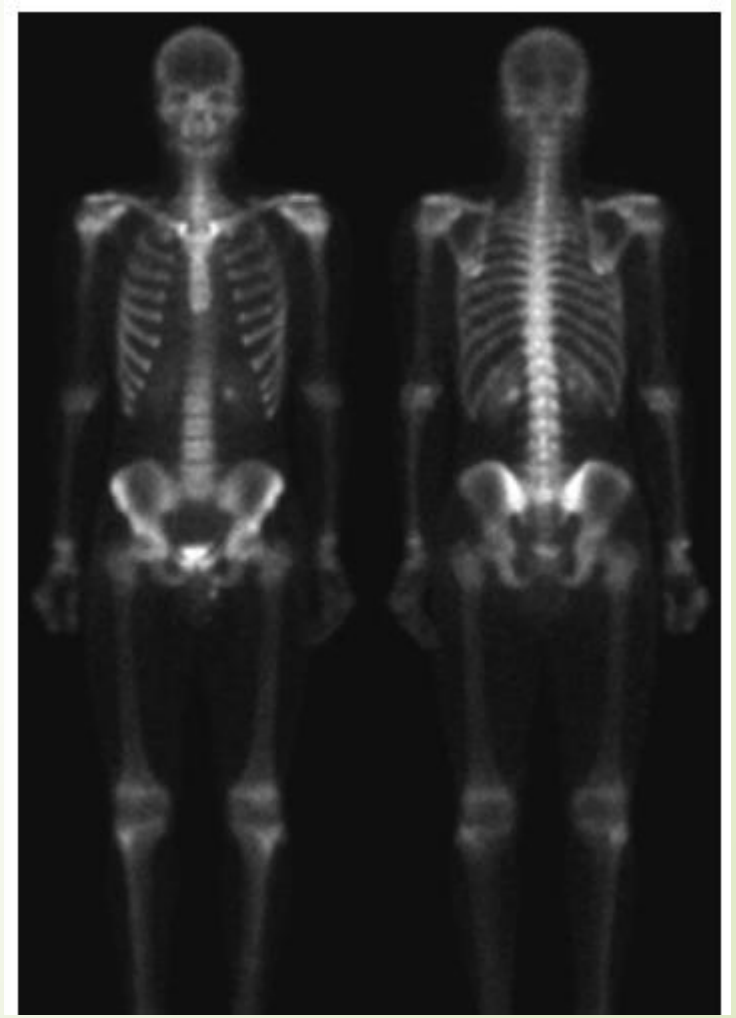
Images based on radiation from the EM spectrum are the most familiar. Electromagnetic waves can be conceptualized as propagating sinusoidal waves of varying ***wavelengths***, or they can be thought of as a stream of massless particles, each traveling in a wavelike pattern and moving at the speed of light. Each massless particle contains a certain amount (or bundle) of energy. Each bundle of energy is called a ***photon***.



Imaging modalities

Gammas-Ray Imaging

Major uses of imaging based on gamma rays include nuclear medicine and astronomical observations. In nuclear medicine, the approach is to inject a patient with a radioactive isotope that emits gamma rays as it decays. Images are produced from the emissions collected by gamma ray detectors.



Imaging modalities

X-Ray Imaging

X-rays for medical and industrial imaging are generated using an X-ray tube, which is a vacuum tube with a cathode and anode. The cathode is heated, causing free electrons to be released. These electrons flow at high speed to the positively charged anode. When the electrons strike a nucleus, energy is released in the form of X-ray radiation.

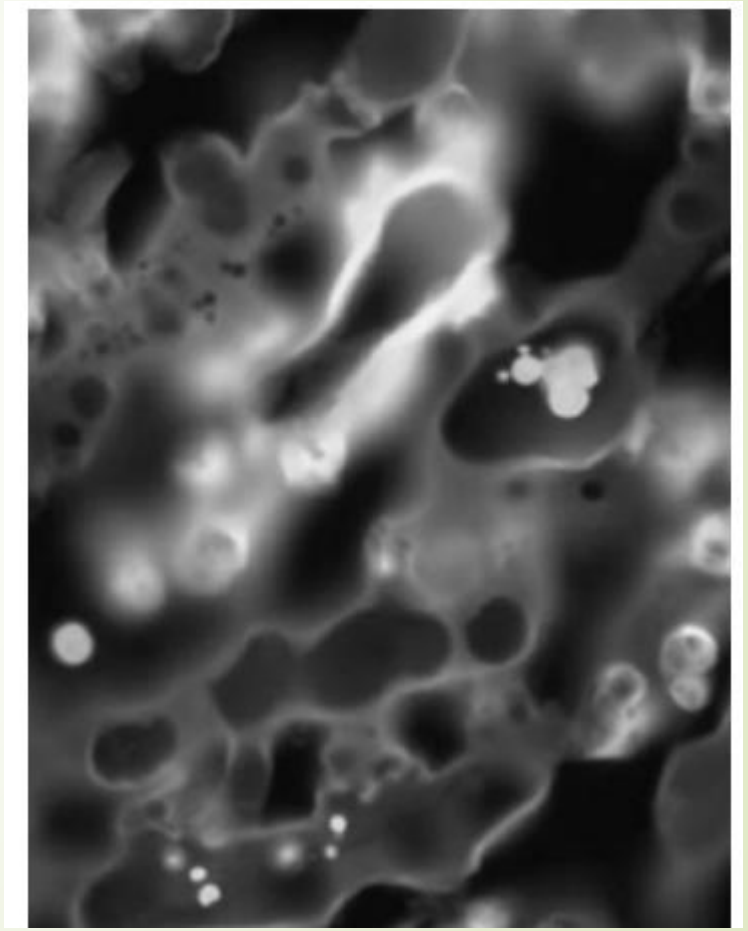
our ordinary medical radios are in X-Ray, which can surpass everything in human body except bones. The shows a familiar chest X-ray generated simply by placing the patient between an X-ray source and a film sensitive to X-ray energy.

Imaging modalities

Ultraviolet Imaging

Ultraviolet light is used in fluorescence microscopy, one of the fastest growing areas of microscopy. Fluorescence is a phenomenon discovered in the middle of the nineteenth century, when it was first observed that the mineral fluorspar fluoresces when ultraviolet light is directed upon it.

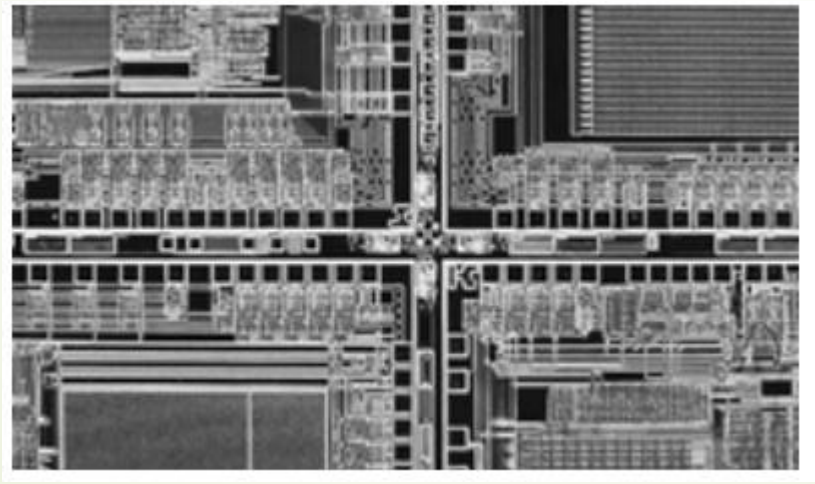
The basic task of the fluorescence microscope is to use an excitation light to irradiate a prepared specimen and then to separate the much weaker radiating fluorescent light from the brighter excitation light. Thus, only the emission light reaches the eye or other detector.



Imaging modalities

Imaging in the Visible and Infrared Bands

The infrared band often is used in conjunction with visual imaging. Images taken using light microscopy are typical examples for images generated using the visual band. light microscopy simply enlarging species being imaged using lens and take photos in presence of light



Imaging modalities

Imaging in the Visible and Infrared Bands

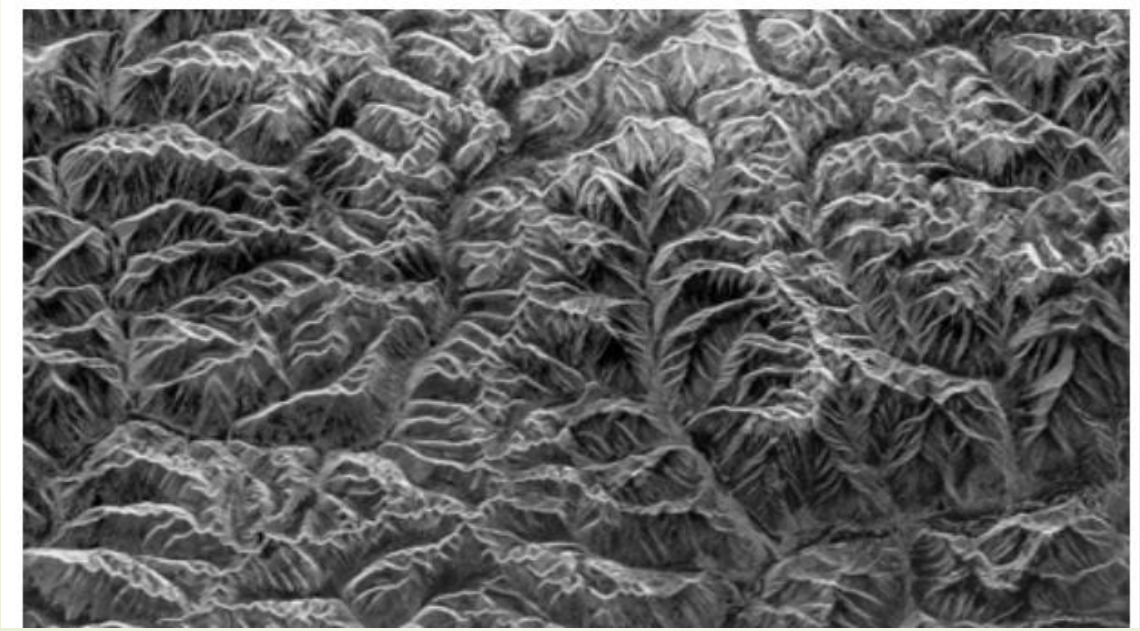
The infrared band often is used in conjunction with visual imaging. Here, an example of infrared image taken using NASA satellite. The Table represents different bands for imaging in this satellite.

Band No.	Name	Wavelength (μm)	Characteristics and Uses
1	Visible blue	0.45–0.52	Maximum water penetration
2	Visible green	0.52–0.60	Good for measuring plant vigor
3	Visible red	0.63–0.69	Vegetation discrimination
4	Near infrared	0.76–0.90	Biomass and shoreline mapping
5	Middle infrared	1.55–1.75	Moisture content of soil and vegetation
6	Thermal infrared	10.4–12.5	Soil moisture; thermal mapping
7	Middle infrared	2.08–2.35	Mineral mapping

Imaging modalities

Microwave Band Imaging

The dominant application of imaging in the microwave band is radar. The unique feature of imaging radar is its ability to collect data over virtually any region at any time, regardless of weather or ambient lighting conditions. The figure is for mountains in southeast Tibet.



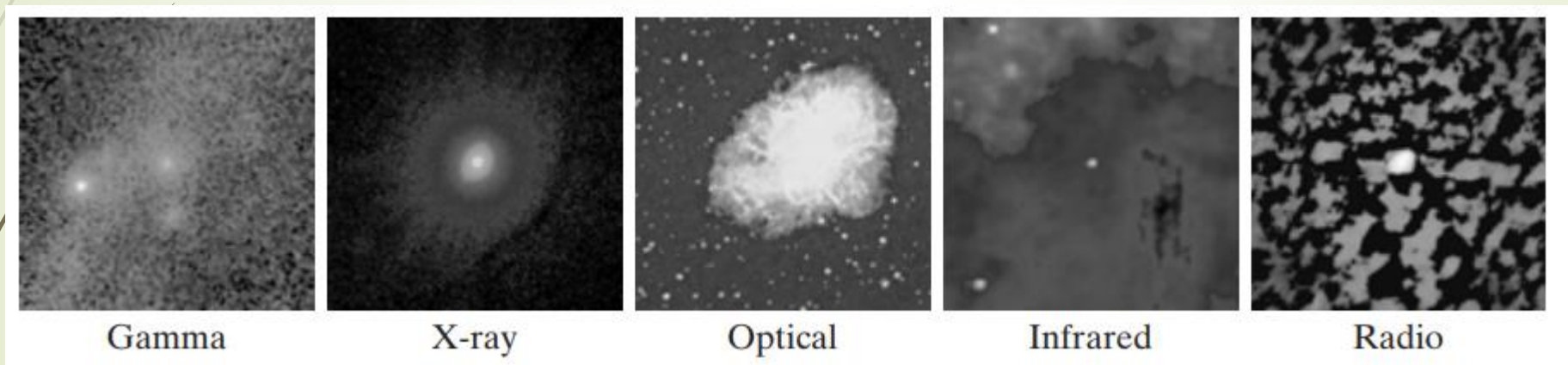
Imaging modalities

Radio Band Imaging

In medicine radio waves are used in magnetic resonance imaging (MRI). This technique places a patient in a powerful magnet and passes radio waves through his or her body in short pulses. Each pulse causes a responding pulse of radio waves to be emitted by the patient's tissues. The location from which these signals originate and their strength are determined by a computer, which produces a two-dimensional picture of a section of the patient.

Imaging modalities

Multispectral image for Crab Pulsar



human visual perception

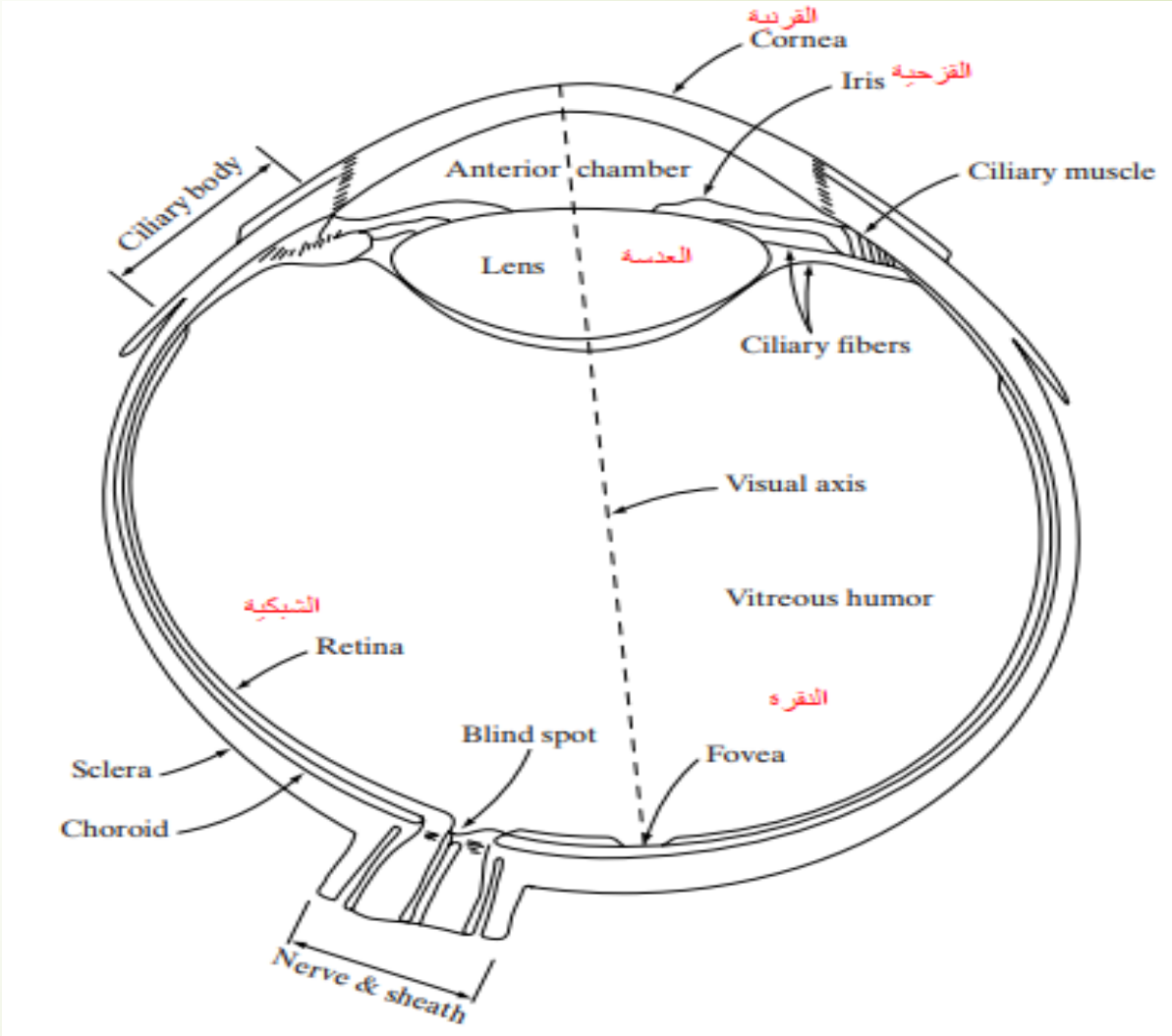
The organ responsible on human visual perception is the eye. Here is a cross diagram for the human eye .

- ***sclera***. contains a network of blood vessels that serve as the major source of nutrition to the eye

- ***Iris*** contracts or expands to control the amount of light that enters the eye.

- **The fovea** is responsible for sharp central vision, which is necessary in humans for activities for which visual detail is of primary importance e.g., reading

- When the eye is properly focused, light from an object outside the eye is imaged on the ***retina***. Pattern vision is afforded by the distribution of discrete light receptors over the surface of the retina.

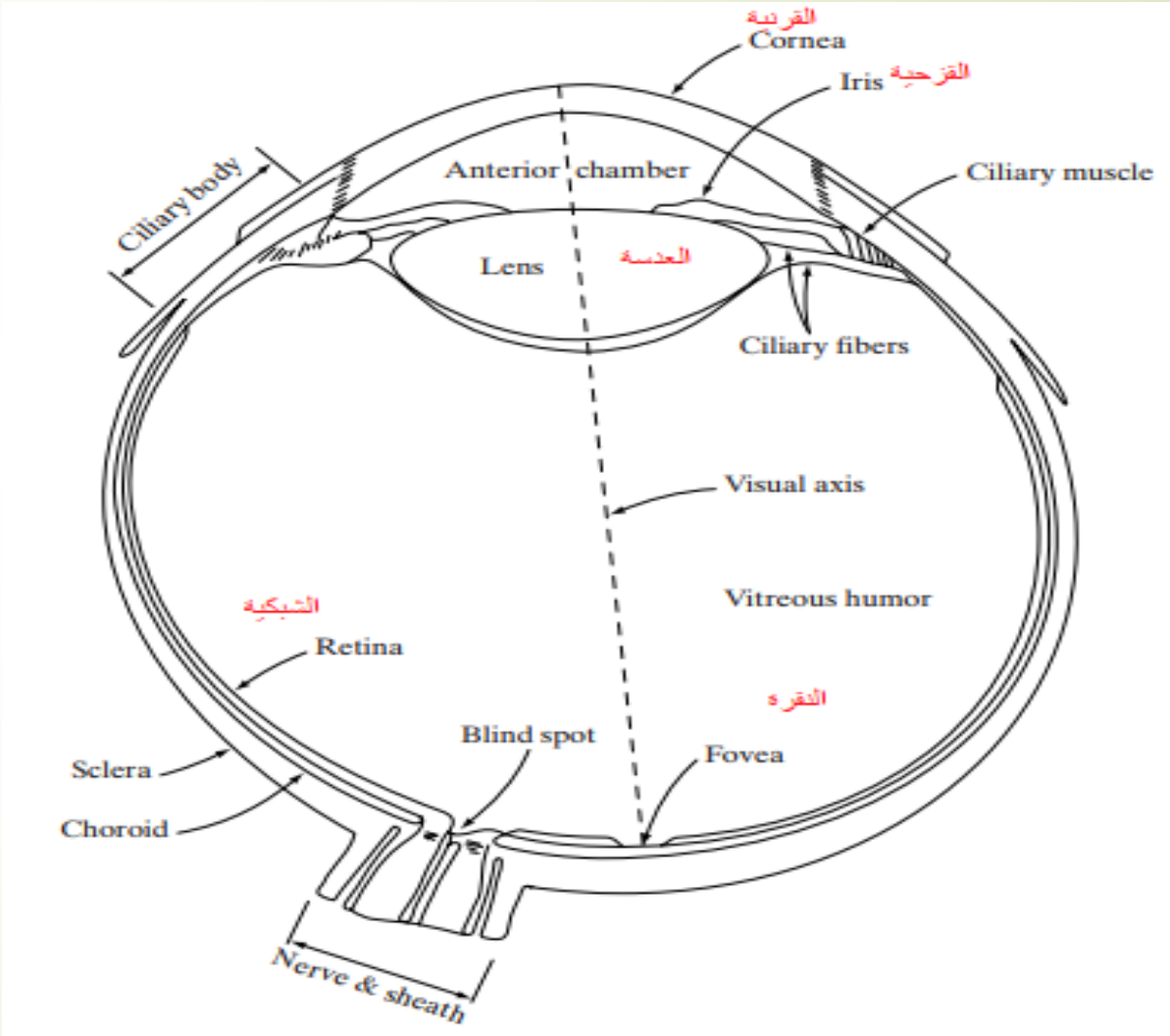


human visual perception

There are two classes of receptors: ***cones*** and ***rods***. The cones in each eye number between 6 and 7 million. They are located primarily in the central portion of the retina, called the ***fovea***.

Humans ***can resolve fine details with these cones*** largely because each one is connected to its own nerve end.

The number of rods is much larger: Some 75 to 150 million are distributed over the retinal surface. The larger area of distribution and the fact that several rods are connected to a single nerve end reduce the amount of detail discernible by these receptors. ***Rods serve to give a general, overall picture of the field of view.***



human visual perception

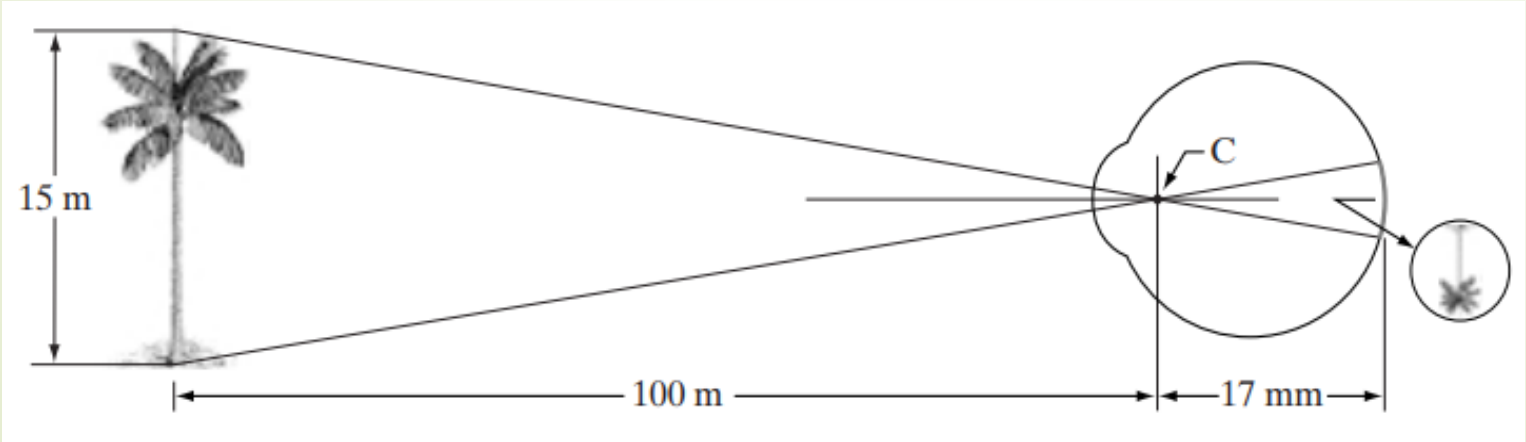
How does eye work?

Alhacen Iben Alhaithem was the first who revealed how human eyes work. This achievement is not the only provided by the Arabic and Islamic civilization in the middle centuries!

human visual perception

How does eye work?

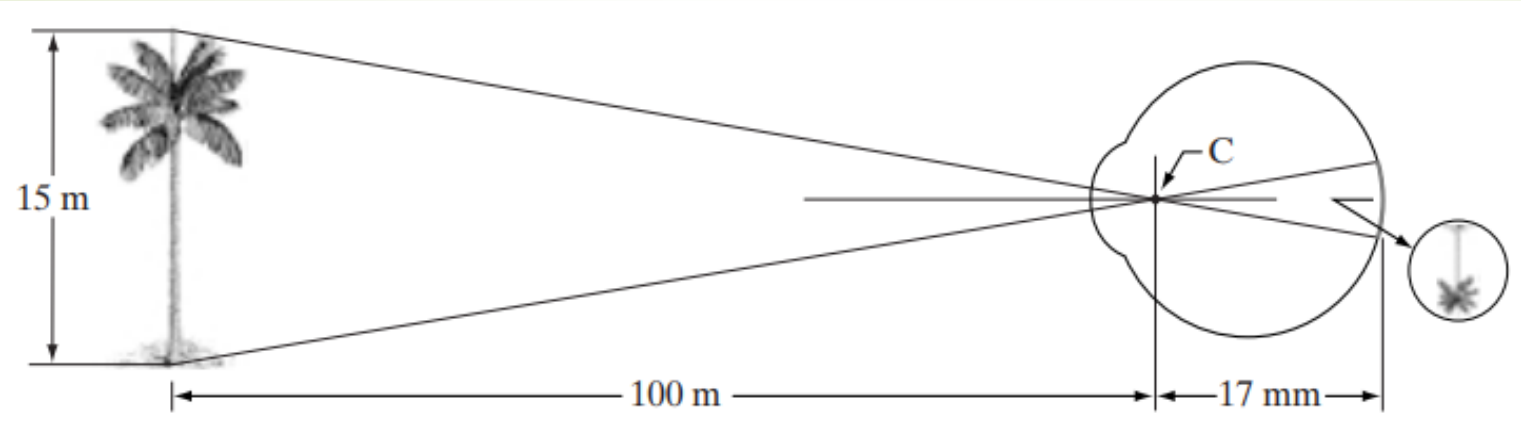
A light beam enter the eye through the cornea, this beam is refrected, as it falls on a convex surface. The image being observed is reflcted on the retina which contains millions of light sensors. After having receving this light, it is sent to the brain as electrical impulses that are ultimately decoded by the brain. We can calculate the size of the retinal image according to the geometry shown by following figure



human visual perception

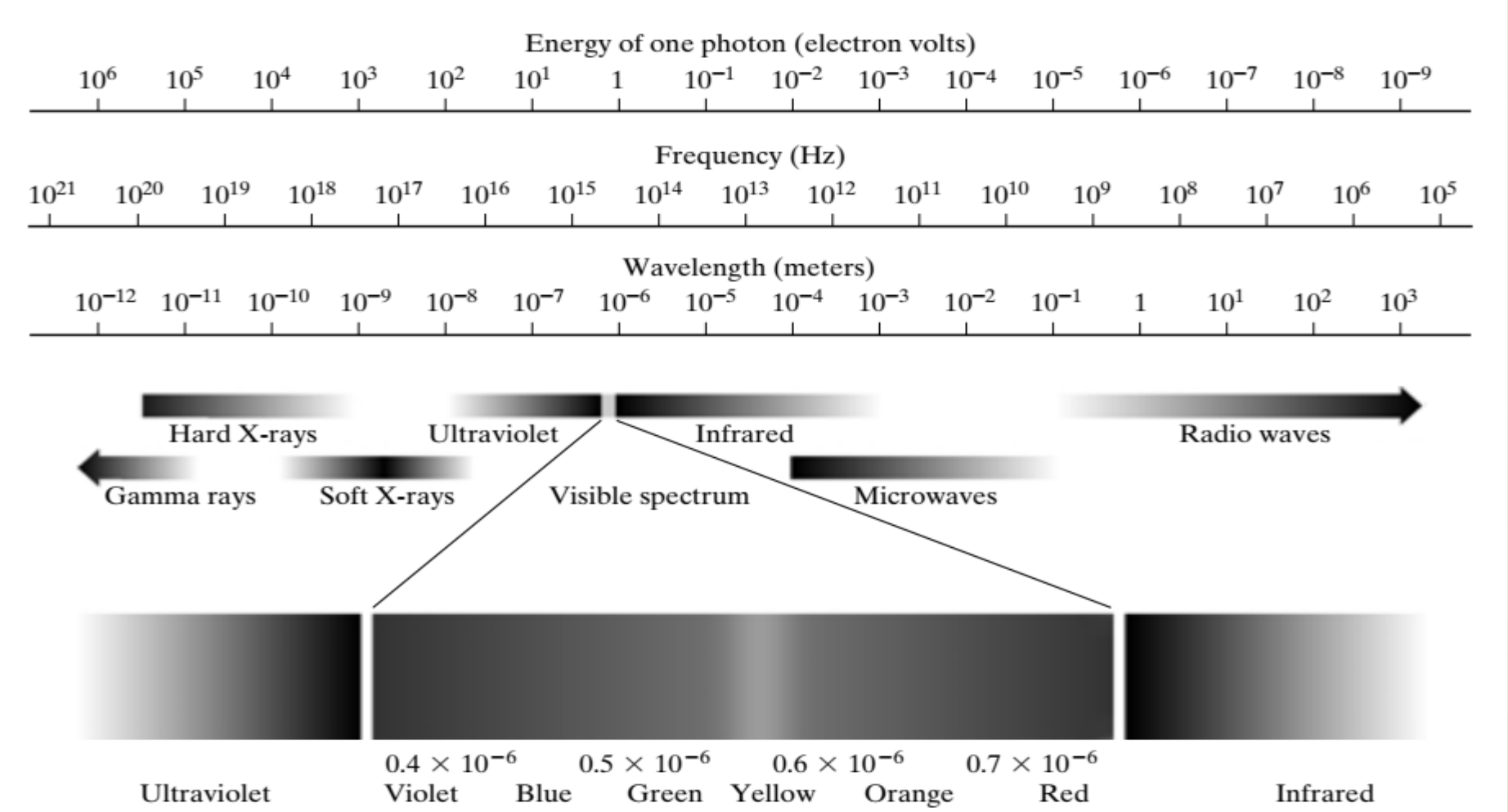
How does eye work?

For example, the observer is looking at a tree 15 m high at a distance of 100 m. If ***h*** is the height in mm of that object in the retinal image, the geometry yields 15/100=h/17 or ***h*=2.55 mm**.



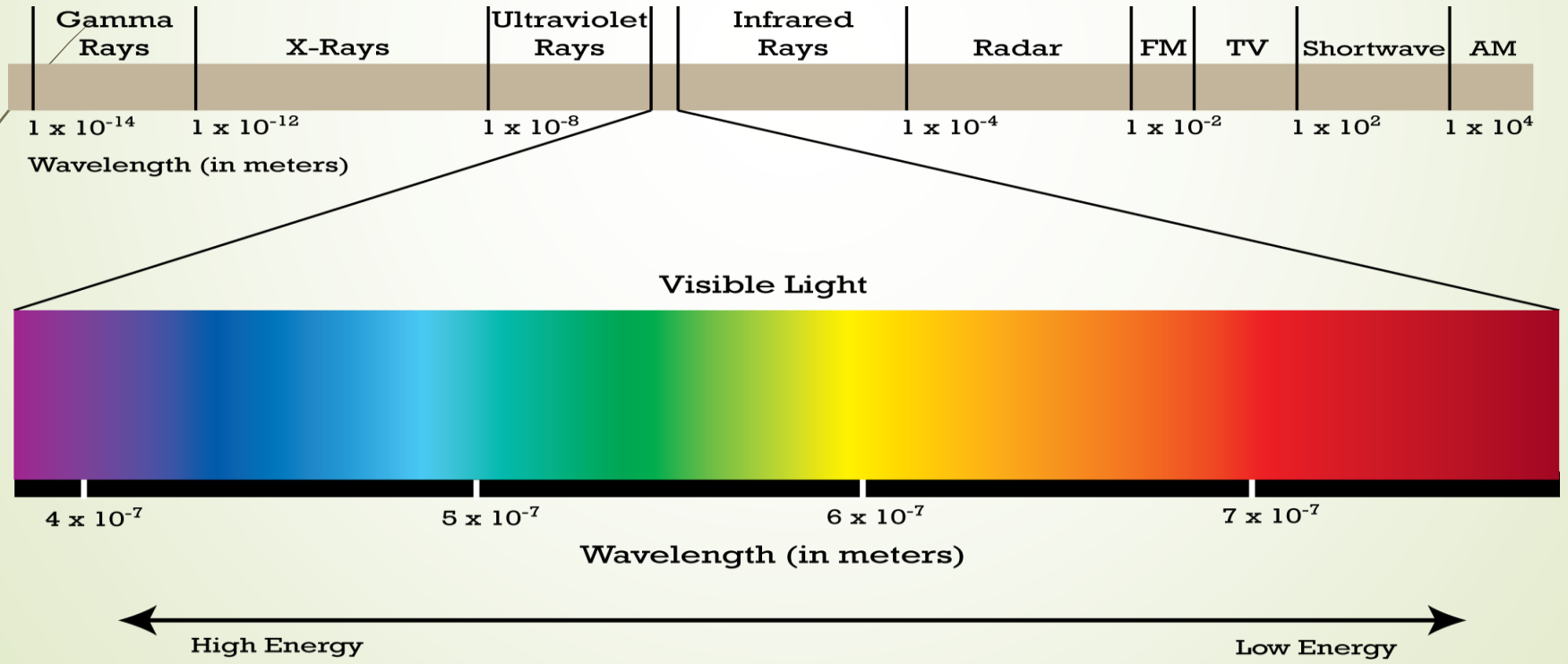
Electromagnetic Spectrum

As it has already mentioned, electromagnetic spectrum can be viewed as massless particles or waves.



Electromagnetic Spectrum

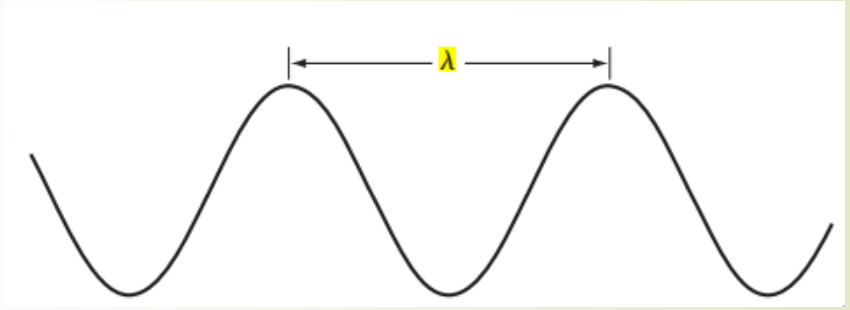
In 1666, Sir Isaac Newton discovered that when a beam of sunlight is passed through a glass prism, the emerging beam of light is not white but consists instead of a continuous spectrum of colors ranging from violet at one end to red at the other.



Electromagnetic Spectrum

The electromagnetic spectrum can be expressed in terms of wavelength, frequency, or energy. Wavelength λ and frequency v are related by the expression

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



Such that c is the speed of light $(2{,}998 \times 10^{8} m/s)$

The energy of the various components of the electromagnetic spectrum is given by the expression

$$E = hv$$

Where h is Planck's constant, The units of wavelength are meters, with the terms *microns* (denoted μm and equal to 10^{-6} m) and *nanometers* (10^{-9} m) being used just as frequently. Frequency is measured in Hertz (Hz), with one Hertz being equal to one cycle of a sinusoidal wave per second. A commonly used unit of energy is the electron-volt.

Electromagnetic Spectrum

Imaging requirement

It is important to note, however, that the wavelength of an electromagnetic wave required to "see" an object must be of the same size as or smaller than the object. For example, a water molecule has a diameter on the order of 10^{-10} m. Thus, to study molecules, we would need a source capable of emitting in the far hard X-ray region. This limitation, along with the physical properties of the sensor material, establishes the fundamental limits on the capability of imaging sensors, such as visible, infrared, and other sensors in use today.

Image Acquisition

Again, it should be noted that Alhacen Iben Alhaithem was the first to invent the camera 'القمرة'

اخترع ابن الهيثم الكاميرا و هي غرفة مظلمة ينفذ اليها الضوء عبر ثقب صغيرة
بتصغير الثقب الى حد كبير تحدث ظاهرة التشتت بحيث بتشتت الشعاع الداخل الى الغرفة ليظهر الصورة مقلوبة
كما أوضح أيضا أن جودة الصورة تزيد كلما صغر الثقب

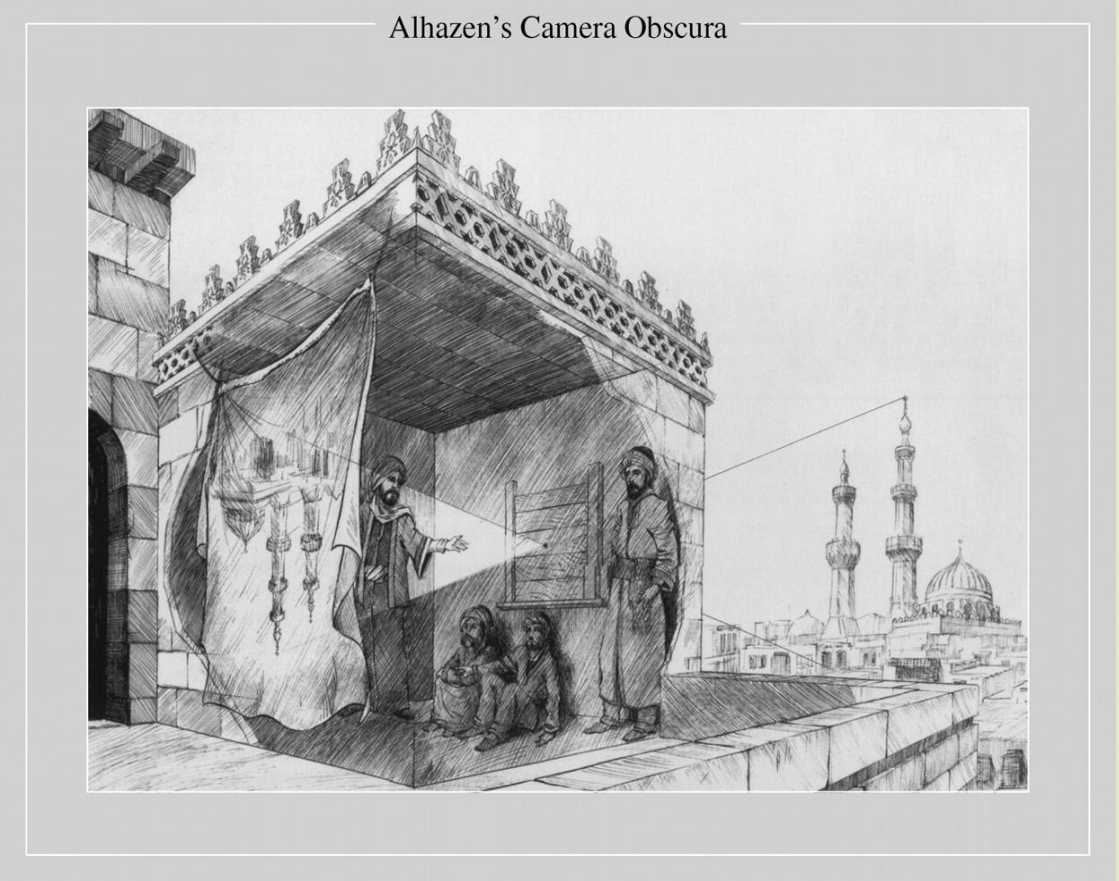


Image Acquisition

Acquisition techniques can be classified according to the industrial technology used for constructing the sensor. In this course, and since numerous electromagnetic and sensing devices frequently are arranged in an array format, we consider sensor arrays

A typical sensor for these cameras is a CCD array, which can be manufactured with a broad range of sensing properties and can be packaged in arrays of 4000 x 4000 elements or more.

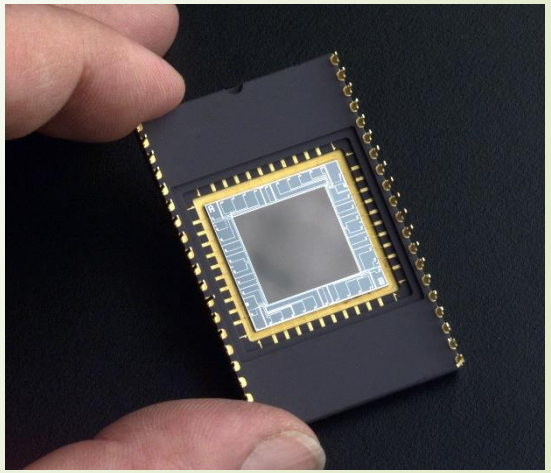


Image Acquisition

The first function performed by the imaging system is to collect the incoming energy and focus it onto an image plane. If the illumination is light, the front end of the imaging system is a lens, which projects the viewed scene onto the lens focal plane. The sensor array, which is coincident with the focal plane, produces outputs proportional to the integral of the light received at each sensor. Digital and analog circuitry sweep these outputs and convert them to a video signal, which is then digitized by another section of the imaging system.

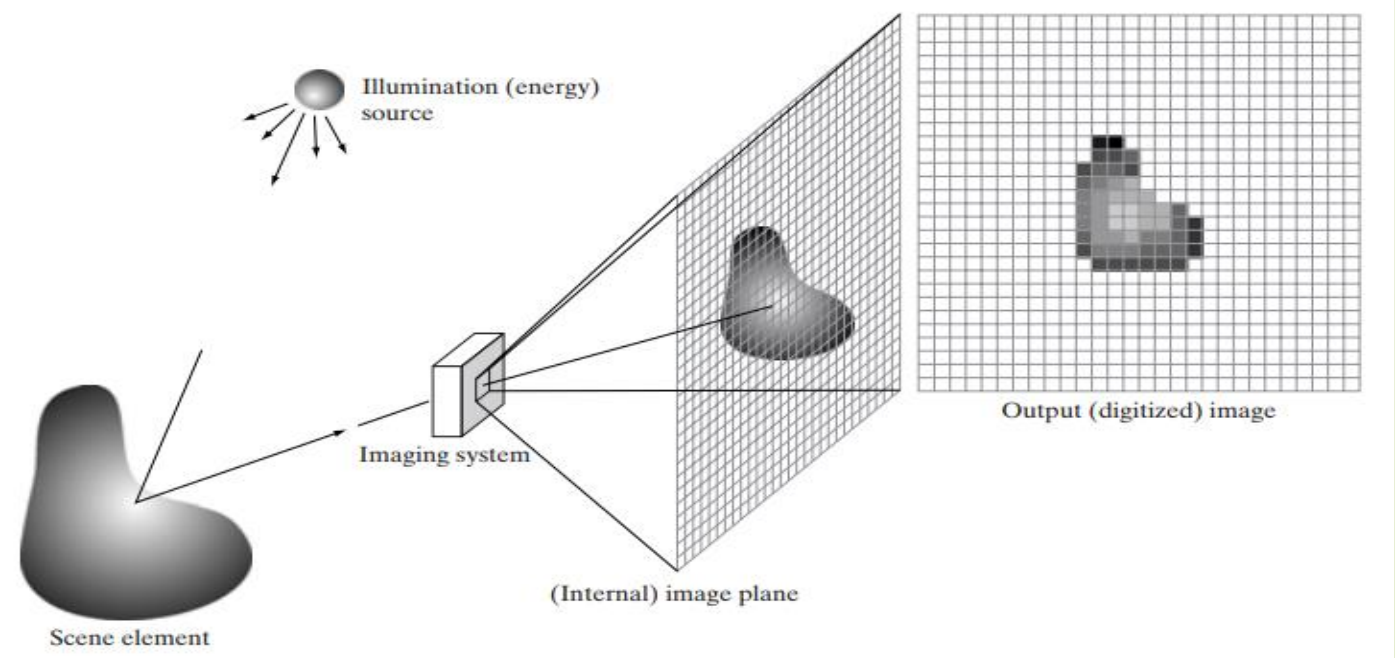


Image Formation

Image is a two-dimensional function of the form **f(x, y).** The value or amplitude of ***f*** at spatial coordinates ***(x, y)*** is a positive scalar quantity. The function ***f(x, y)*** may be characterized by two components: (1) the amount of source illumination incident on the scene being viewed, and (2) the amount of illumination reflected by the objects in the scene. Appropriately, these are called the ***illumination*** and ***reflectance*** components and are denoted by ***i(x, y)*** and ***r(x, y),*** respectively. The two functions combine as a product to form f(x, y):

$$f(x, y) = i(x, y) \cdot r(x, y)$$

Where $0<i(x, y)<\infty$ and $0<r(x, y)<1$

indicates that reflectance is bounded by 0 (total absorption) and 1 (total reflectance).The nature of ***i(x, y)*** is determined by the illumination source, and ***r(x, y)*** is determined by the characteristics of the imaged object.

Image Formation

we call the ***intensity*** of a ***monochrome*** image at any coordinates **(x,y)** the ***gray level (L)*** of the image at that point

$$L = f(x,y)$$

where ***L*** falls in the range $L_{min} < L < L_{max}$

The interval is called the *gray scale*. Common practice is to shift this interval numerically to the interval ***[0, L-1],*** where ***L=0*** is considered black and ***L=L-1*** is considered white on the gray scale. All intermediate values are shades of gray varying from black to white.

Image Sampling and Quantization

An image may be continuous with respect to the x- and y-coordinates, and also in amplitude. To convert it to digital form, we have to sample the function in both coordinates and in amplitude. Digitizing the coordinate values is called ***sampling***. Digitizing the amplitude values is called ***quantization***. The one-dimensional function shown in Figure is a plot of amplitude (gray level) values of the continuous image along the line segment ***AB***. To sample this function, we take equally spaced samples along line ***AB***. In order to form a digital function, the gray-level values also must be converted (***quantized***) into discrete quantities.

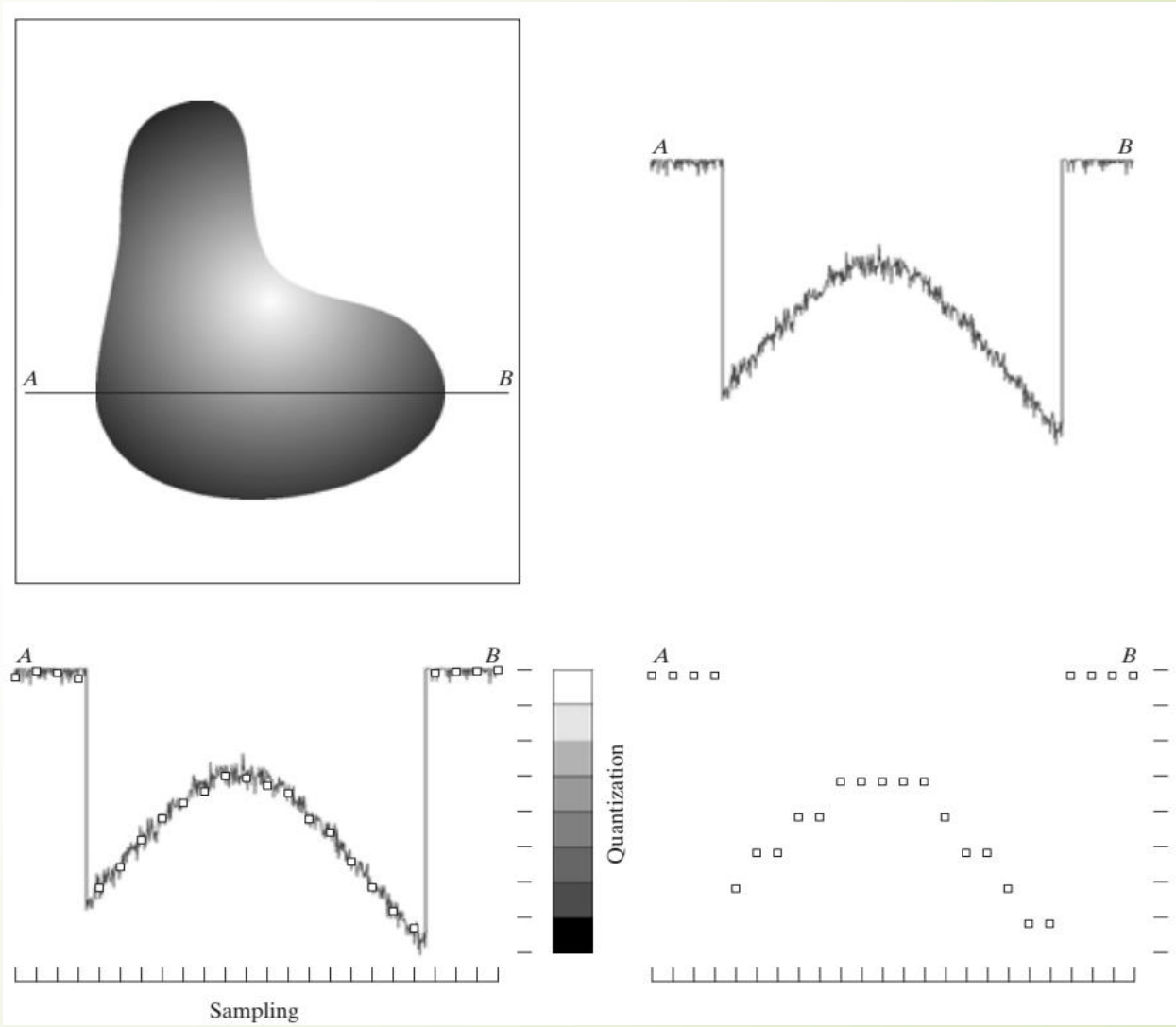
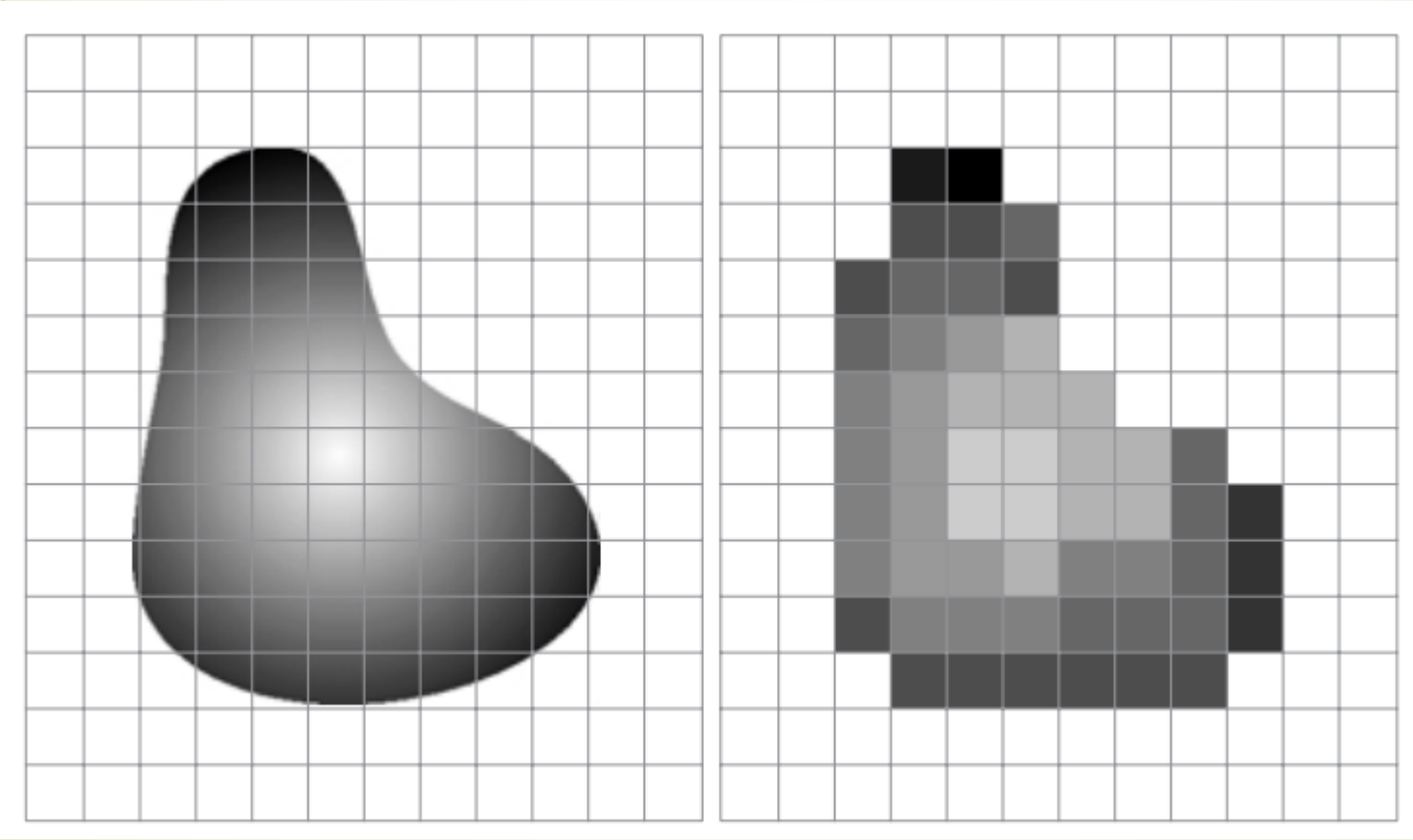


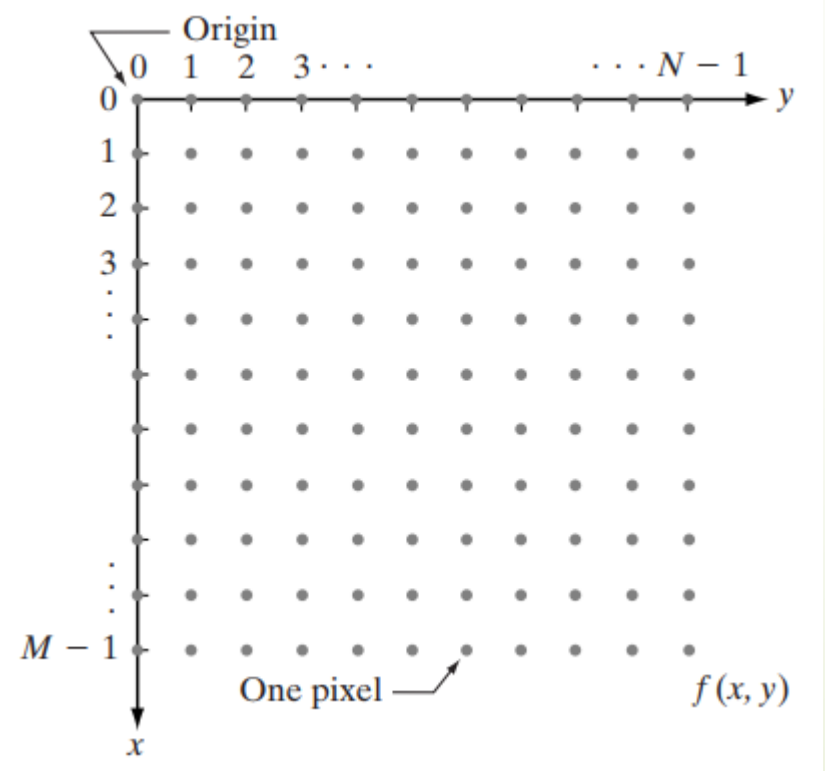
Image Sampling and Quantization

The final output looks like



Representation of Digital Image

Assume that an image ***f(x, y)*** is sampled so that the resulting digital image has ***M*** rows and ***N*** columns. The values of the coordinates ***(x, y)*** now become ***discrete*** quantities. Thus, the values of the coordinates at the origin are ***(x, y)=(0, 0).*** The next coordinate values along the first row of the image are represented as ***(x, y)=(0, 1).***



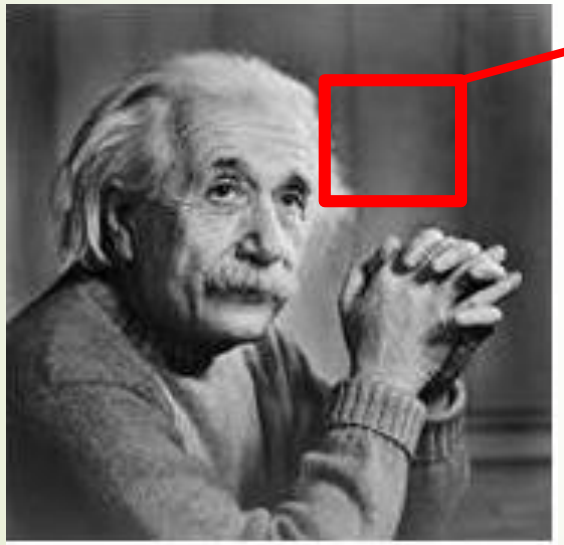
Representation of Digital Image

The notation introduced in the preceding paragraph allows us to write the complete ***MxN*** digital image in the following matrix form

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

Each element of this matrix array is called an ***image element***, ***picture element, pixel***

Representation of Digital Image



255	100	0	5	22	10
147	20	128	26	78	99
33	25	58	0	0	11
37	221	111	52	68	98
255	87	102	84	5	7
44	78	158	255	0	0

Brightness and contrast

Brightness is a relative term! For two different images we may judge which one is brighter (Resp. darker).

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Image 1

0,3	0,3	0,3	0,3
0,3	0,3	0,3	0,3
0,3	0,3	0,3	0,3
0,3	0,3	0,3	0,3

Image 2 = Image 1+0,3

0,6	0,6	0,6	0,6
0,6	0,6	0,6	0,6
0,6	0,6	0,6	0,6
0,6	0,6	0,6	0,6

Image 3 = Image 2+0,3

Brightness and contrast

We notice that image 2 is brighter than image 1. Contrary, we can see that image 2 is darker than image 3. It can be calculated according the following equation (average brightness)

$$B = \frac{1}{M \times N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} I(i,j)$$

As for contrast, it can be defined as the difference between maximum and minimum pixel intensity in an image.

$$\boldsymbol{Contrast} = Max\ intensity\ - Min\ intensity$$

Example: contrast of image 1 = contrast of image 2 = 0

Brightness and contrast

Other definitions for the contrast

$$\sqrt{\frac{1}{MN}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}\left(I_{ij}-\bar{I}\right)^2}$$

$$\frac{I_{\max}-I_{\min}}{I_{\max}+I_{\min}}$$

Low contrast image

High contrast image

Representation of Digital Image

How many level there are?

This digitization process requires decisions about values for ***M***, ***N***, and for the number, ***L***, of discrete gray levels allowed for each pixel. There are no requirements on ***M*** and ***N***, other than that they have to be positive integers. However, due to processing, storage, and sampling hardware considerations, the number of gray levels typically is an integer power of **2**

$$L = 2^K$$

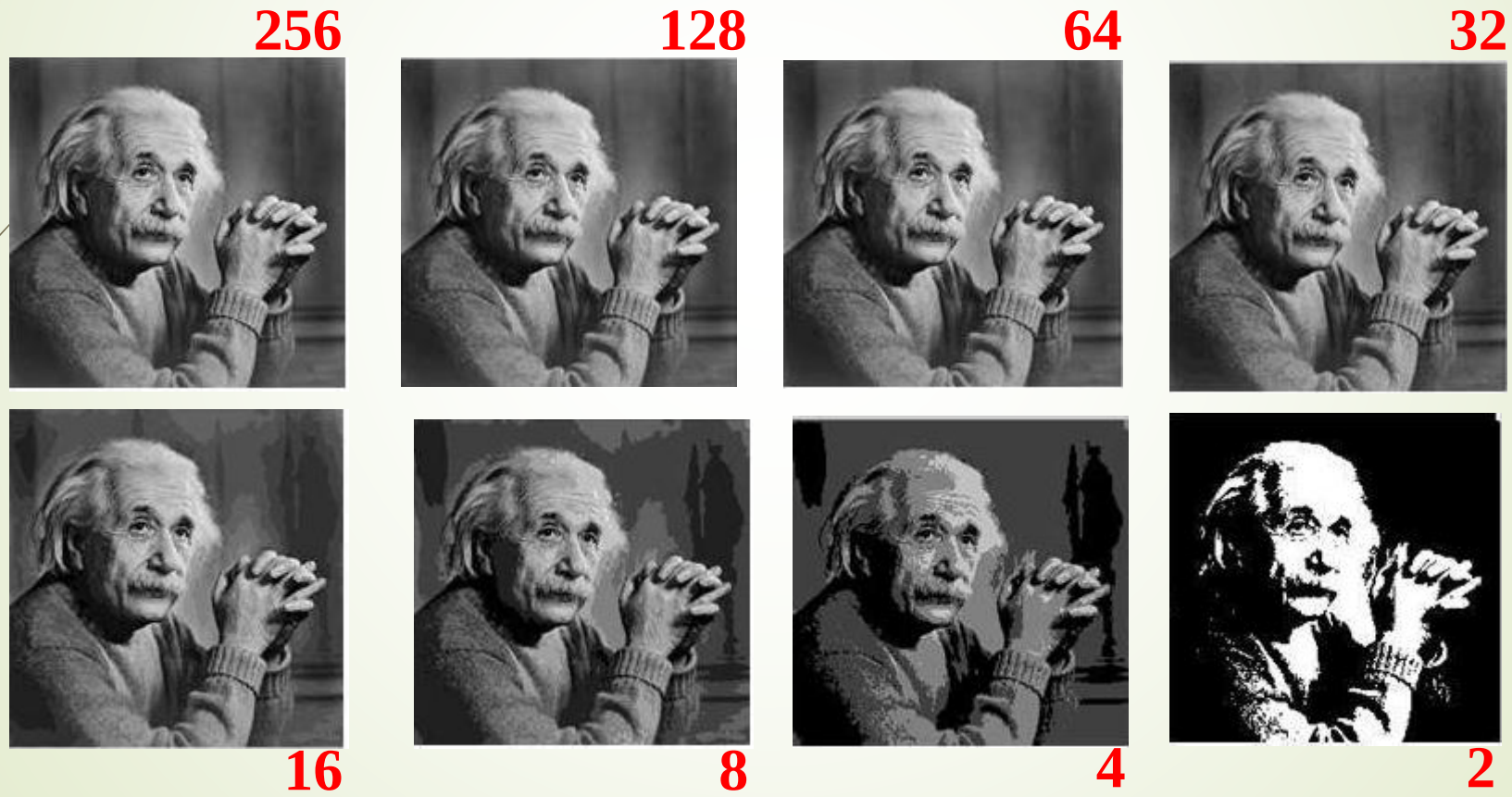
K in this case is called **B**its **P**er **P**ixel (**BPP**) or ***bit depth***, which refers to the number of bits dedicated to represent intensity of a given pixel.

Generally, in gray-level images ***BPP = 8***. Thus, the total number of levels is 2^8***=256***

00000000
00000001
00000010
00000011
……
……
……
11111111

Representation of Digital Image

The following image shows the effect of reducing number of gray level progressively



Storage Requirements

Theoretically, the number, ***b***, of bits required to store a digitized image is

$$b = M \times N \times K$$

N/k	1 ($L = 2$)	2 ($L = 4$)	3 ($L = 8$)	4 ($L = 16$)	5 ($L = 32$)	6 ($L = 64$)	7 ($L = 128$)	8 ($L = 256$)
32	1,024	2,048	3,072	4,096	5,120	6,144	7,168	8,192
64	4,096	8,192	12,288	16,384	20,480	24,576	28,672	32,768
128	16,384	32,768	49,152	65,536	81,920	98,304	114,688	131,072
256	65,536	131,072	196,608	262,144	327,680	393,216	458,752	524,288
512	262,144	524,288	786,432	1,048,576	1,310,720	1,572,864	1,835,008	2,097,152
1024	1,048,576	2,097,152	3,145,728	4,194,304	5,242,880	6,291,456	7,340,032	8,388,608
2048	4,194,304	8,388,608	12,582,912	16,777,216	20,971,520	25,165,824	29,369,128	33,554,432
4096	16,777,216	33,554,432	50,331,648	67,108,864	83,886,080	100,663,296	117,440,512	134,217,728
8192	67,108,864	134,217,728	201,326,592	268,435,456	335,544,320	402,653,184	469,762,048	536,870,912

For an image of **300 x 500** pixels ***Size= Height*** × ***Width*** × ***BPP = 300*** × ***500*** × ***8 = 12 00 000 bit***

Size (Ko) = (12 00 000 / 8) / 1024 = 146 Ko

Storage Requirements

Calculate the size of color image and its gray level version

If for each channel RGB and gray scale, we have a bit depth = 8, and if the image size is **300 x 500** pixels then,

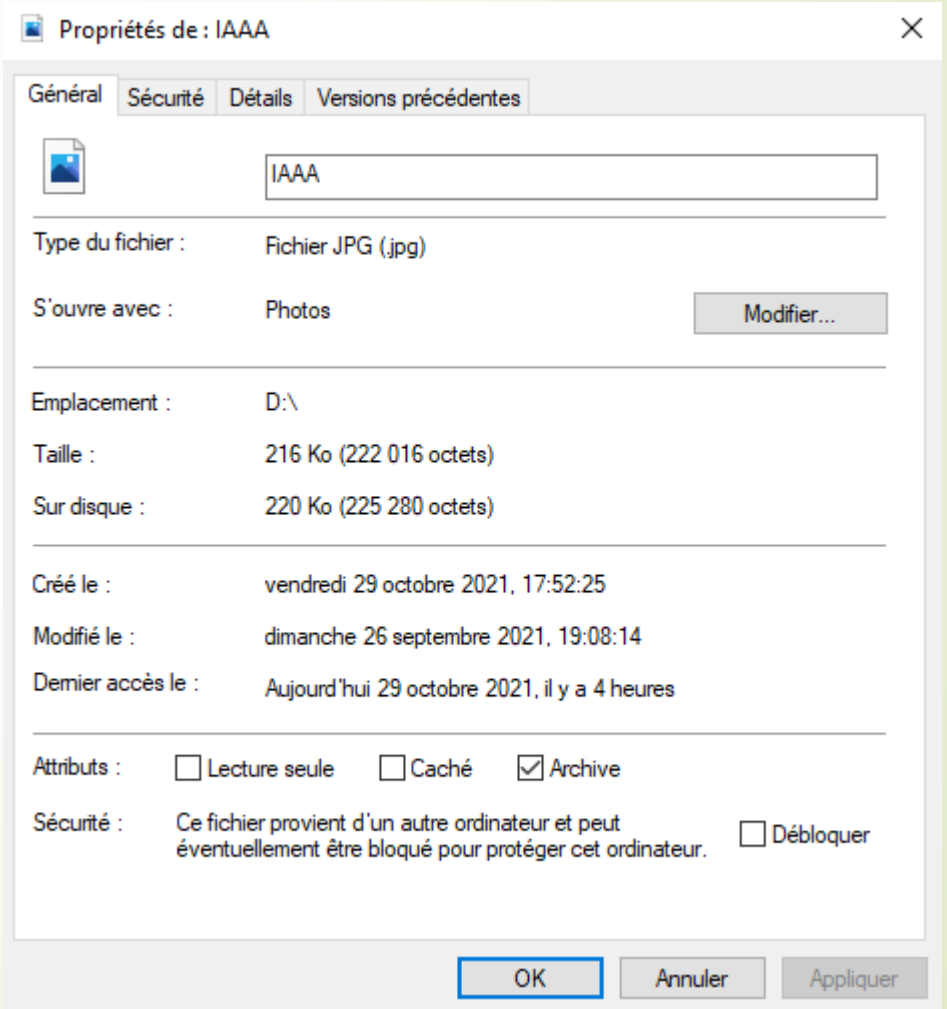
Size of Gray image= 300 × 500 × 8 = 12 00 000 bits = (12 00 000 / 8) / 1024 = 146 Ko

Size of Color image= 300 × 500 × 24 = 36 00 000 bits = (36 00 000 / 8) / 1024 = 439 Ko

Image formats

Actual VS calculated image size

*Calculated size = **439 Ko***
*Actual size = **216 Ko !!!!! why***



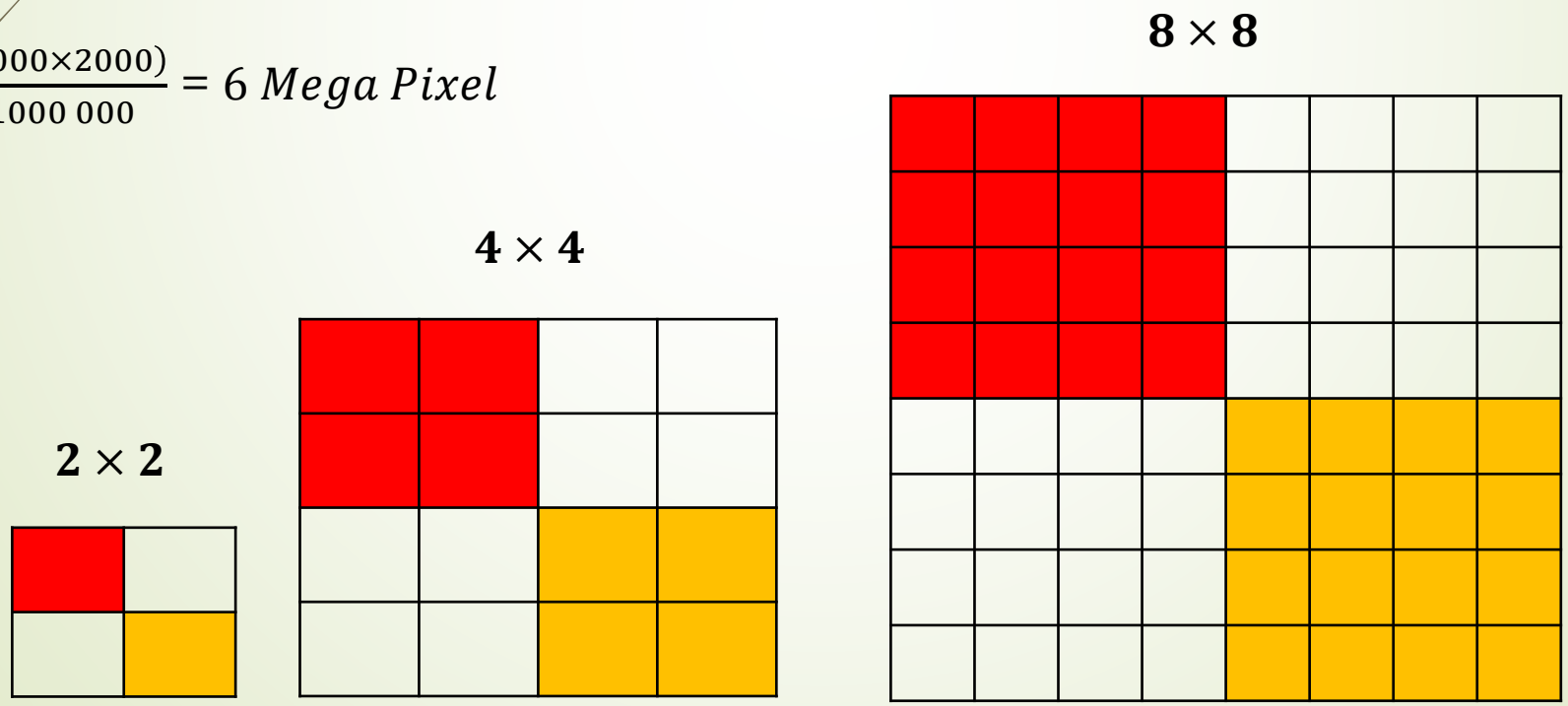
Pixel and Spatial Resolution

It refers to the total number of pixels in the image. Given an image I with a height = M and width = N

Pixel Resolution $(\boldsymbol{I}) = M \times N$

In ***Mega Pixels,*** pixel resolution is $\frac{M \times N}{1\,000\,000}$ E.g., for an image of 3000×2000 dimensions, pixel resolution is equal to

$$\frac{(3000 \times 2000)}{1000\,000} = 6\ Mega\ Pixel$$



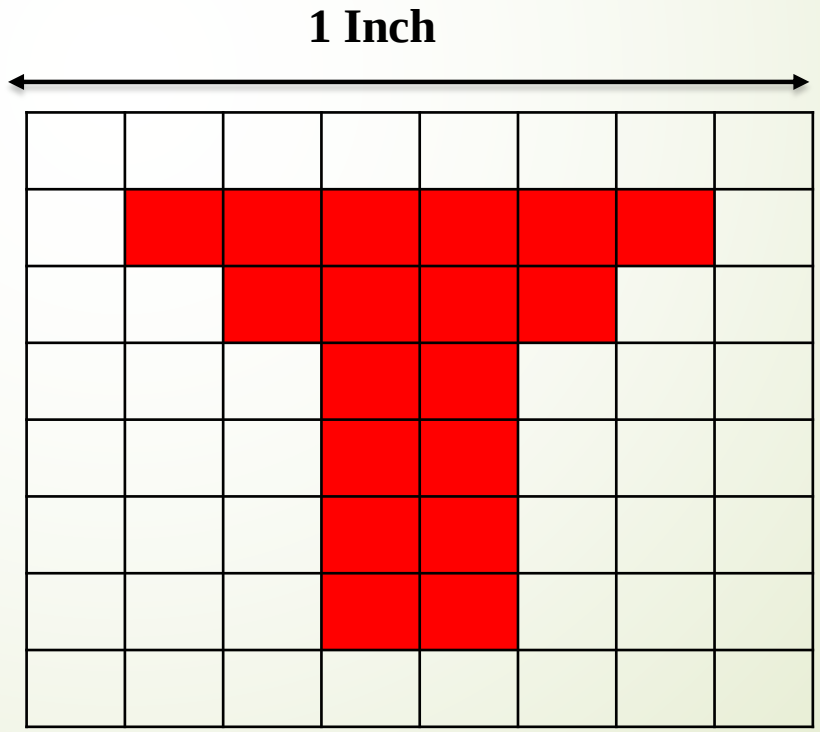
Pixel and Spatial Resolution

What is the difference between the two following images

Pixel and Spatial Resolution

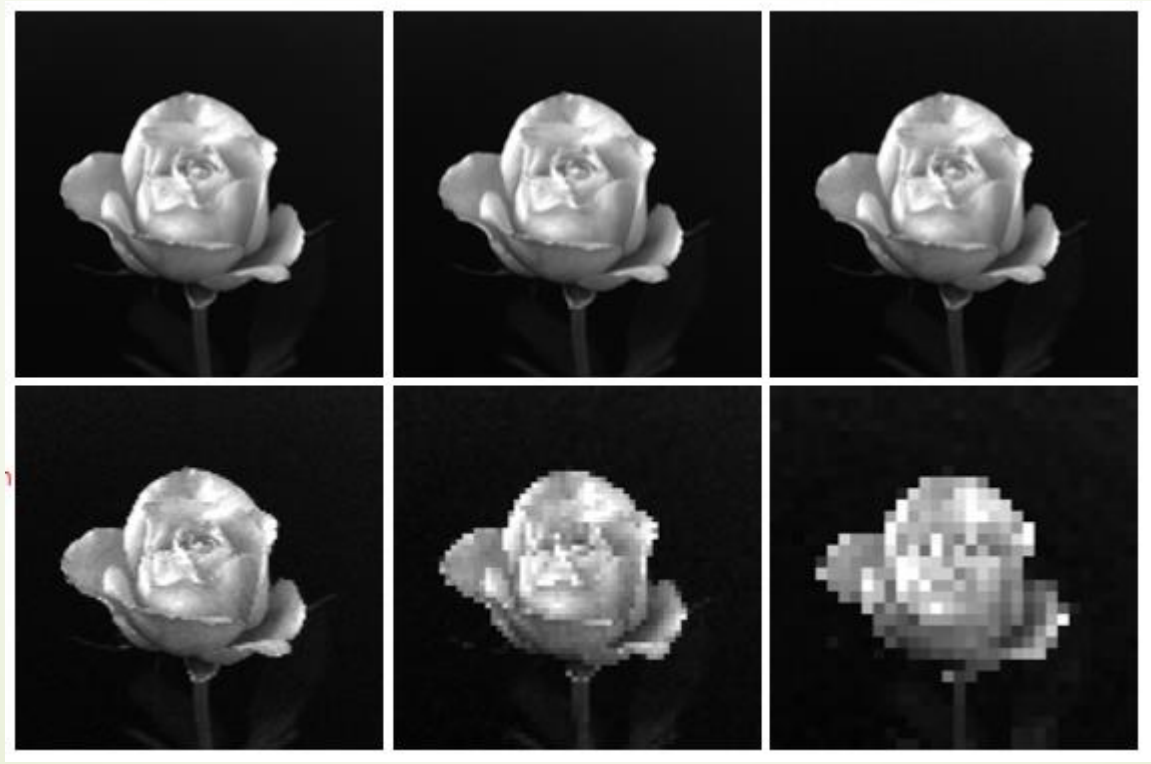
Suppose that we construct a chart with vertical lines of width ***W***, with the space between the lines also having width ***W***. A ***line pair*** consists of one such line and its adjacent space. A widely used definition of resolution is simply the smallest number of discernible line pairs per unit distance; for example, 100 line pairs per millimeter.

Pixel resolution = 8×8
Spatial resolution = 8 pixels/Inch



Pixel and Spatial Resolution

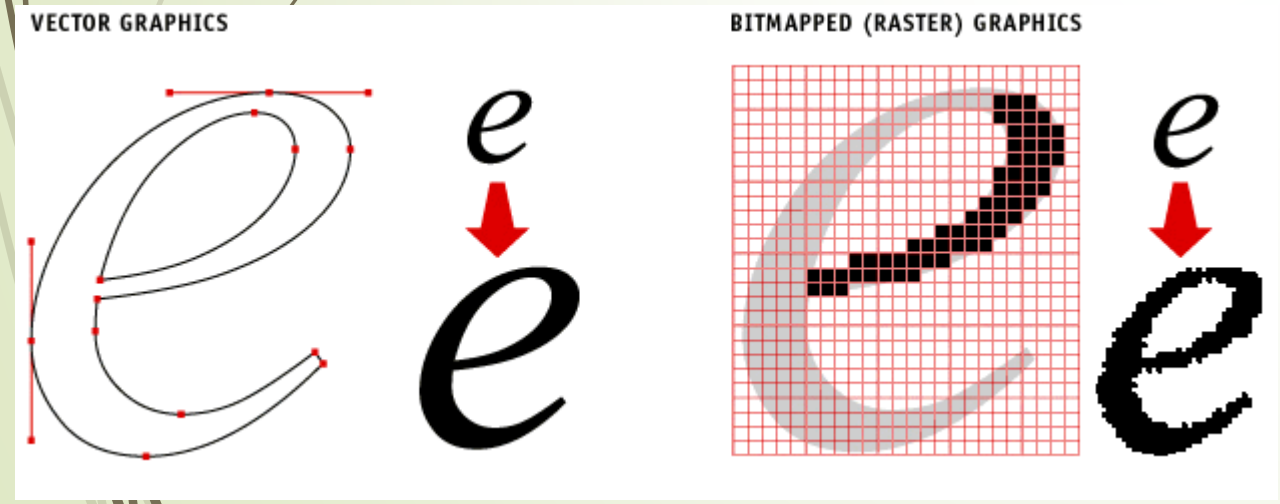
Increasing image size doesn't increase its spatial resolution, noting that increasing size can be done by replicating every row and column

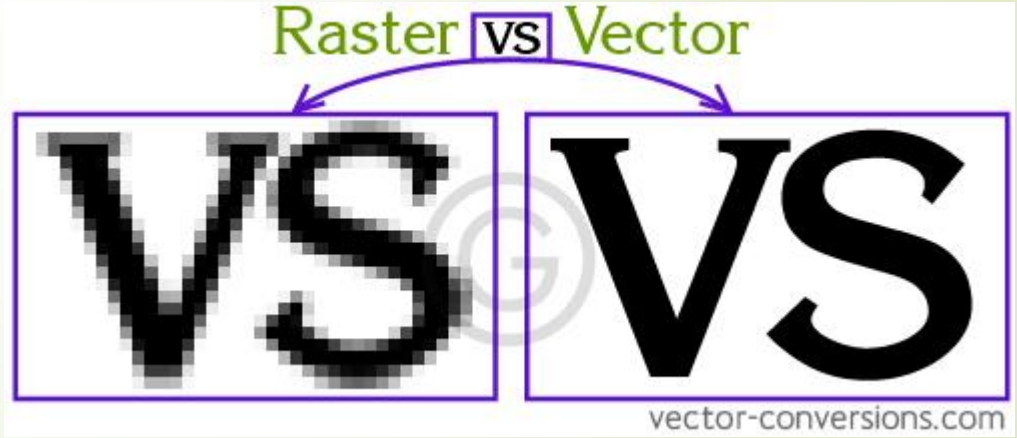


Raster versus vector image

Mainly, there are two type of images namely ***Raster*** and ***Vector***.

Raster images are pixel-based and they are created using pixel-based software or captured with a camera or scanner e.g., jpg, gif, png. Vector images are math-defined shapes created with vector software and are not common (e.g., they are used in graphic Design). Vector images created using geometric shapes defined on a Cartesian plane, such as points, lines, Curves and polygons.





Raster versus vector image

Raster	Vector
Created using pixels	Shapes based on mathematical calculations
Well-suited for editing photos and soft color blends	Well-suited for logos, drawings and illustrations
Scale up image degrade its quality. Image must be created/scanned for a better resolution	Can be scaled to any size without losing quality
Large dimensions = large file size	A large dimension vector graphic maintains a small file size. For example, a square can be unambiguously defined by the locations of three of its four corners only.
Depending on the complexity of the image, conversion to vector may be time consuming	Vector image can be easily rasterized to be used for all processes
The most common image format including: jpg, gif, png, tif, bmp, psd, eps	Common formats: ai, cdr, svg, and eps created by vector programs
Common programs for photo editing and painting such as Photoshop, Paint Shop, GIMP.	Common vector programs: drawing programs such as Illustrator, CorelDraw, Inkscape.

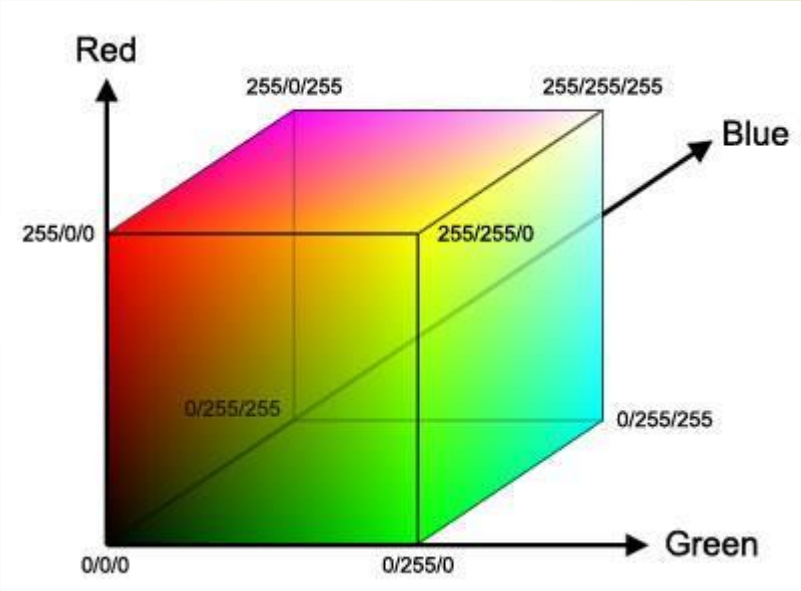
Image formats

There are several image formats which differ from each other in terms of several features, common formats include JPEG, PNG, BMP, GIF, TIF,…..etc.

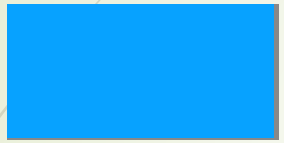
	JPEG	PNG	TIF	PSD	BMP
Max Pixel Depth	8 bits	16 bits	32 bits	16 bits	8 bits
Compression	JPEG	Deflate	None / LZW / RLE / ZIP	NONE / RLE	NONE / RLE
Transparency	NON	YES	YES	YES	YES
Type of image	Raster	Raster	Raster / vector	Raster / vector	Raster

Color Spaces (RGB)

- The principle of this color space is to represent every visible color into a combination of three colors: **R**ed, **G**reen and **B**lue (primary colors).
- Wavelength of these color are $\boldsymbol{\lambda_R = 700\,nm}$ $\boldsymbol{\lambda_G = 546\,nm}$ $\boldsymbol{\lambda_B = 435\,nm}$
- The intensity of each primary color ranges from 0 to 255.
- By mixing different intensities of the primary colors, about 17 000 000 (256^3) colors is counted.



Color Spaces (RGB)



$(7, 162, 255)$

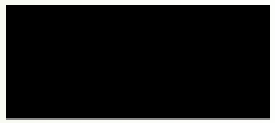
$(140, 255, 7)$

$(140, 255, 30)$

$(188, 61, 15)$

$(71, 101, 223)$

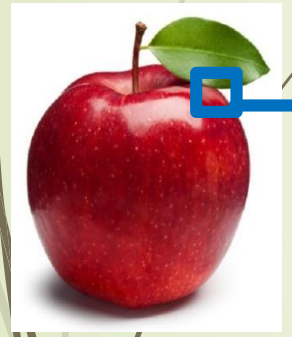
$(255, 255, 255)$



$(0, 0, 0)$

However, it is difficult for humans to distinguish colors having very close values.

Color Spaces (RGB)



255	220	223
58	222	10
78	111	12

25	0	1
25	255	10
100	20	54

125	17	47
0	124	255
0	13	95

R

G

B

Color Spaces (RGB)

Conversion of color image to gray-scale

As gray scale represents one channel, one way to generate intensity image is to average the values of the three RGB channels as follows:

$$I_{GL}(X,Y) = I_R(X,Y) + I_G(X,Y) + I_B(X,Y)$$

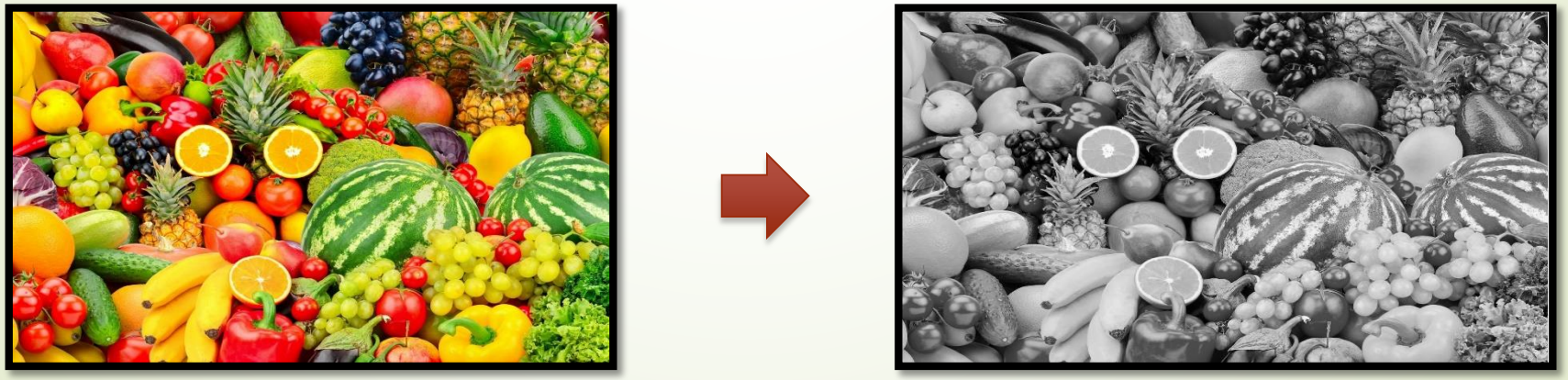
Of course, this is not the gray level image we always see. The resulting image seems much more darker.

Color Spaces (RGB)

Conversion of color image to gray-scale

Through many experiments, psychologists figured out that humans perceive red, green and blue differently e.g., green is look brighter several times than blue. Therefore, a weighted averaging is considered to calculate the intensity image.

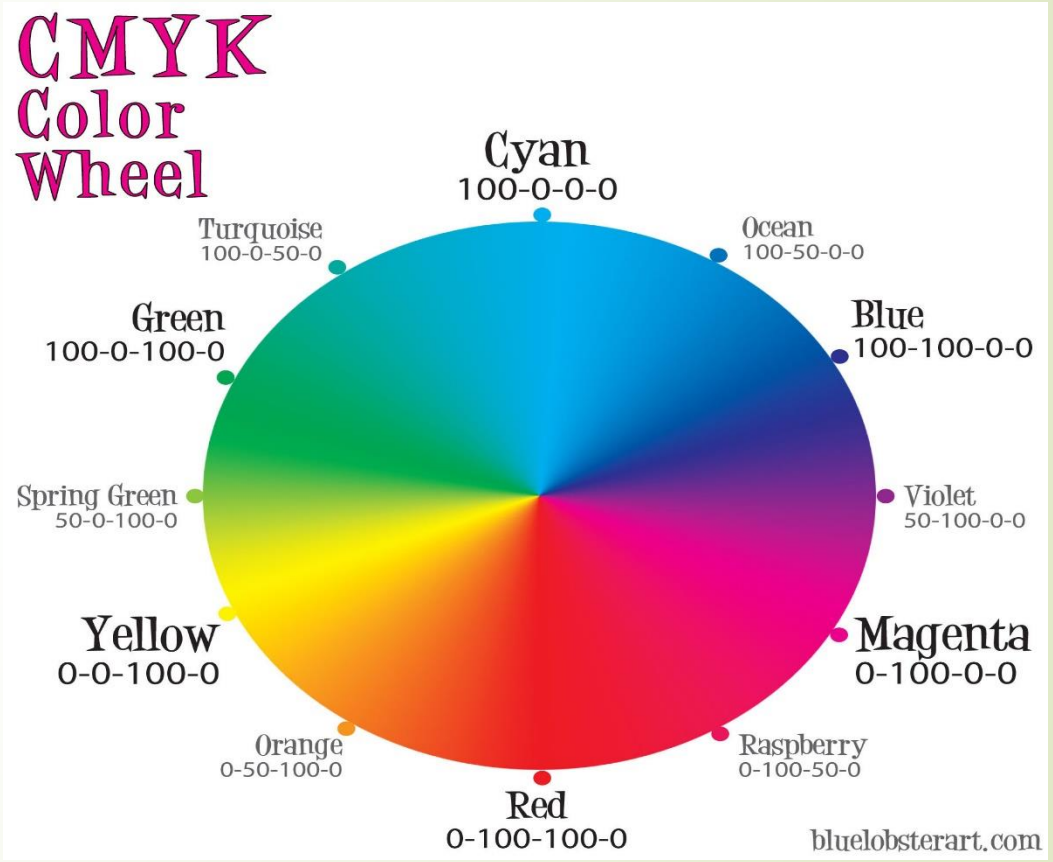
$$I_{GL}(X,Y) = 0.29 \times I_R(X,Y) + 0.58 \times I_G(X,Y) + 0.11 \times I_B(X,Y)$$



Color Spaces (CMYK)

CMYK stands for **C**yan, **M**agenta, **Y**ellow and **K**ey (i.e., black), and it is used by printers. Compared to RGB, which uses light to show images in different screens (TV, PC, smartphone,…), CMYK uses colored ***ink*** to mask colors on a light background (white paper). This light background (usually white) reflects light, so each layer of ink applied subtracts from white light to make new colors.

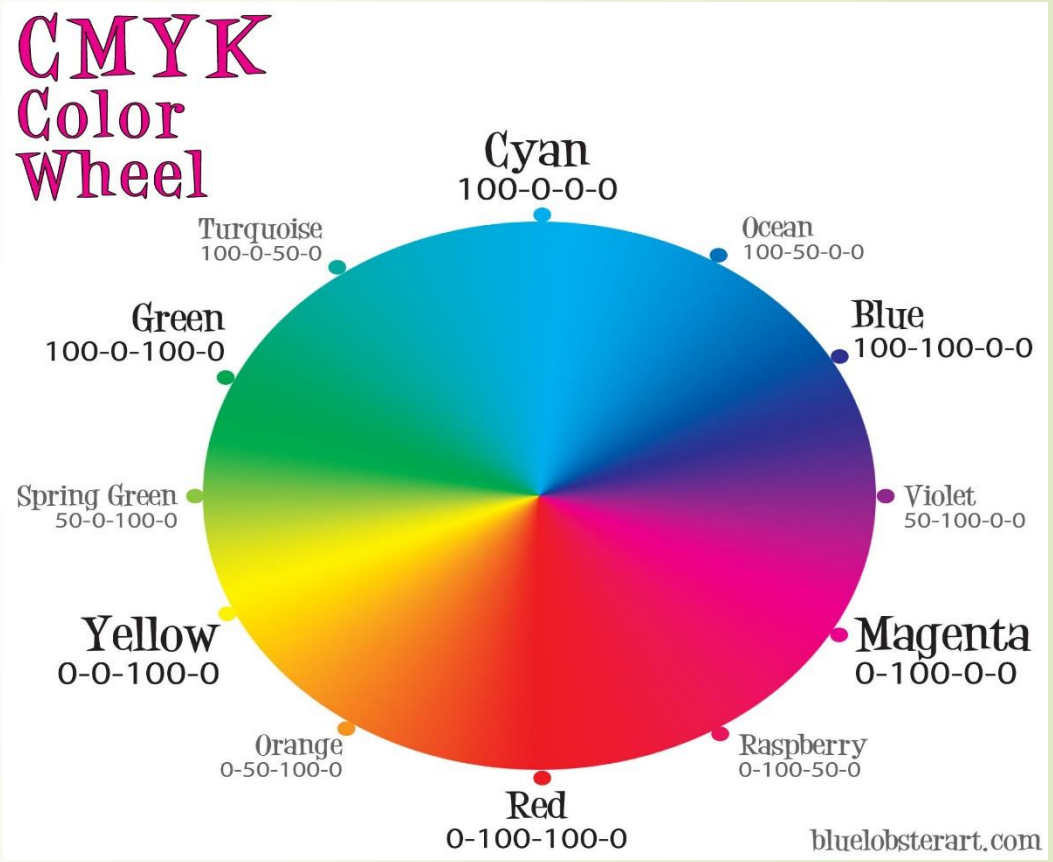
The CMYK color model is a ***subtractive*** color model. ***Subtractive*** means that the more ink added, the closer you get to black. In the contrary, RGB is ***additive*** model because we add colors to each other to go from black to white.



Color Spaces (CMYK)

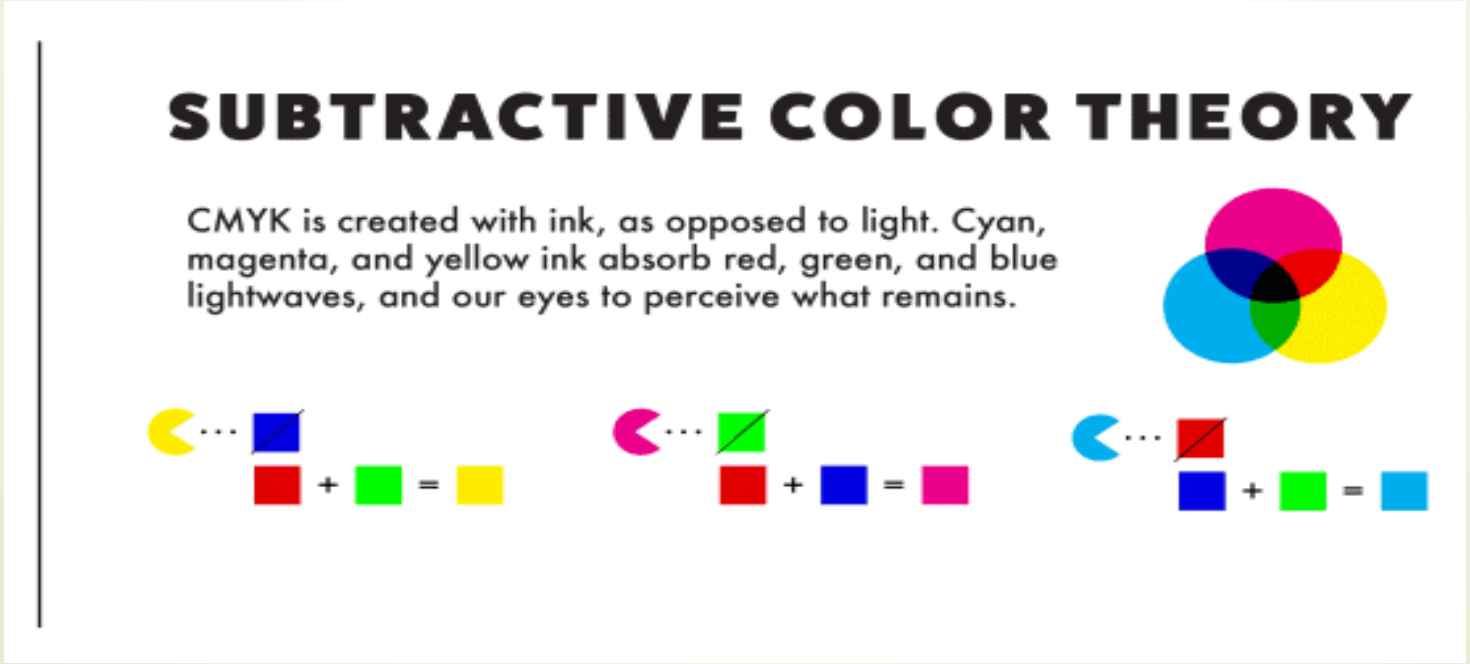
Why CMYK is used for printing

Compared to RGB which starts from black color and end up with white, in CMYK, all colors start as white (e.g., white paper), and each layer of ink reduces the initial brightness to create the preferred color. Thus, since the paper on which we print is white, CMYK is used for printing.



Color Spaces (CMYK)

Printers use ***cyan***, ***magenta*** and ***yellow*** inks in various percentages to control the amount of ***red***, ***green*** and ***blue*** light reflected from ***white*** paper. ***Cyan*** is the complement of ***red*** i.e., the ***cyan*** serves as a filter that absorbs ***red***. The amount of ***cyan*** applied to a ***white*** sheet of paper controls how much of the ***red*** in white light will be reflected back from the paper. It is worth mentioning that the ***cyan*** is completely transparent to ***green*** and ***blue*** light and has no effect on those parts of the spectrum. ***Magenta*** is the complement of ***green***, and ***yellow*** the complement of ***blue***.



Color Spaces (CMYK)

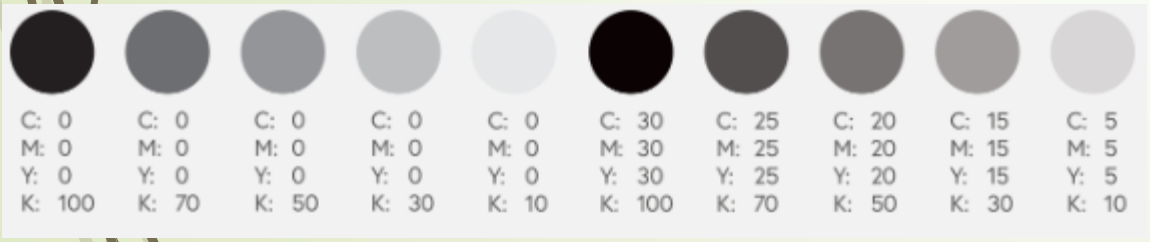
Here, we can notice that to produce ***red*** color, the value of cyan should be 0, as ***cyan*** controls the amount of ***red***. To produce the ***Green*** color the value of ***Magenta*** should be 0. In addition, to produce ***Blue*** the value of ***Yellow*** should be 0.

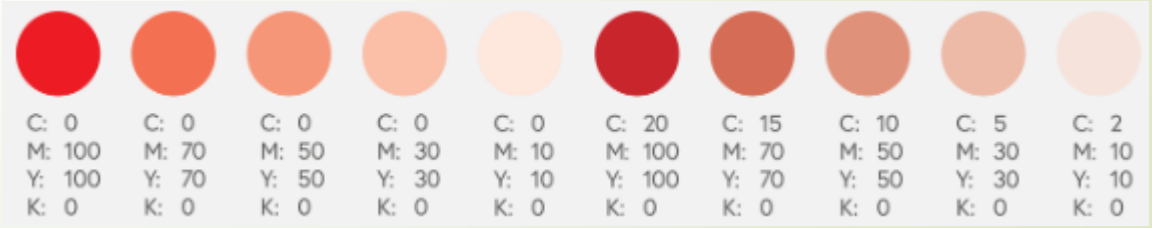
Note that pure black cannot be produced by mixing up the three components (Magenta, Cyan, Blue), for this reason a fourth component is added (key or black).

Color	Color name	(R,G,B)	Hex	(C,M,Y,K)
	Black	(0,0,0)	#000000	(0,0,0,1)
	White	(255,255,255)	#FFFFFF	(0,0,0,0)
	Red	(255,0,0)	#FF0000	(0,1,1,0)
	Green	(0,255,0)	#00FF00	(1,0,1,0)
	Blue	(0,0,255)	#0000FF	(1,1,0,0)
	Yellow	(255,255,0)	#FFFF00	(0,0,1,0)
	Cyan	(0,255,255)	#00FFFF	(1,0,0,0)
	Magenta	(255,0,255)	#FF00FF	(0,1,0,0)

Color Spaces (CMYK)

The values of CMYK are ranging from 0 to 100. **0 0 0 0** gives white (No ink is supplied), whereas, **0 0 0 100** gives black color.





Color Spaces (CMYK)

Conversion from RGB to CMYK

The R,G,B values are divided by 255 to change the range from 0/255 to 0/1

$$R' = R/255$$

$$G' = G/255$$

$$B' = \mathrm{B}/255$$

$$K = 1\text{-}\max(R', G', B')$$

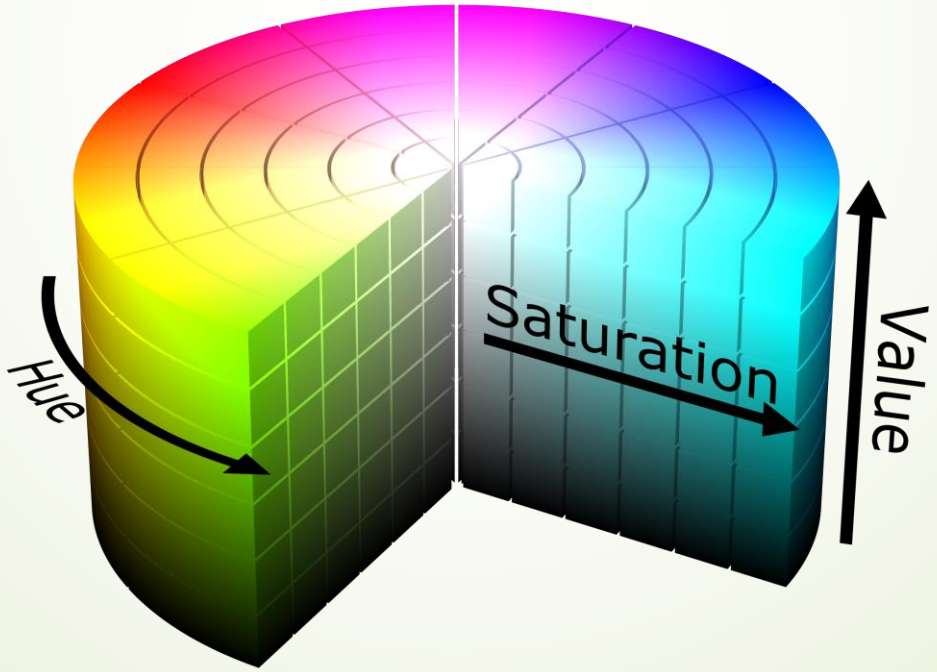
$$C = (1\text{-}R'\text{-}K) \,/\, (1\text{-}K)$$

$$M = (1\text{-}G'\text{-}K) \,/\, (1\text{-}K)$$

$$Y = (1\text{-}B'\text{-}K) \,/\, (1\text{-}K)$$

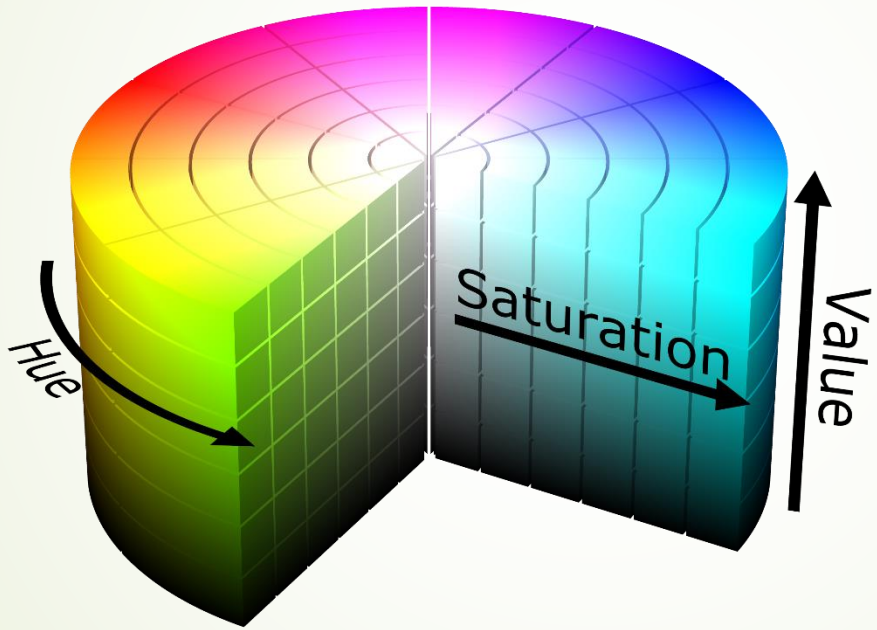
Color Spaces (HSV)

HSV color space is the most close space to the human perception i.e., how humans perceive colors. Indeed, human perception is not like RGB or CMYK colors. They are just primary colors fused to create the spectrum. The ***H*** stands for ***Hue***, ***S*** stands for ***Saturation***, and the ***V*** stand for ***value***.



Color Spaces (HSV)

Hue	
0-60	Red
60-120	Yellow
120-180	Green
180-240	Cyan
240-300	Blue
300-360	Magenta



Saturation: represents the amount of white light that is mixed with the color. A 100% saturation means stands for the complete pure color, whereas, a 0% saturation means no color (pure white). We can note that for saturation = 0, it's the *gray-scale*.

Value: represents the brightness concerning the saturation of the color. It represents the chromatic notion of intensity. The lower the value is the similar to black is (all colors (Hue) will be similar to black). The value 0 represents total black, while the value 100 will mean a full brightness and depend on the saturation.

Saturation and value are well-suited to represent the notion of color brightness and darkness, for this reason HSV is considered to be close to the human perception

Color Spaces (HSV)

Conversion from RGB to HSV

$$R' = R/255$$

$$G' = G/255$$

$$B' = B/255$$

$$Cmax = \max(R', G', B')$$

$$Cmin = \min(R', G', B')$$

$$\Delta = Cmax - Cmin$$

Hue calculation:

$$H = \begin{cases} 0^\circ & \Delta = 0 \\ 60^\circ \times \left(\frac{G'-B'}{\Delta} mod 6\right) & , C_{max} = R' \\ 60^\circ \times \left(\frac{B'-R'}{\Delta} + 2\right) & , C_{max} = G' \\ 60^\circ \times \left(\frac{R'-G'}{\Delta} + 4\right) & , C_{max} = B' \end{cases}$$

Saturation calculation:

$$S = \begin{cases} 0 & , C_{max} = 0 \\ \frac{\Delta}{C_{max}} & , C_{max} \neq 0 \end{cases}$$

Value calculation:

$$V = Cmax$$

Color Spaces (HSV)

Conversion from HSV to RGB

When $0 \le H < 360$, $0 \le S \le 1$ and $0 \le V \le 1$:

$$C = V \times S$$

$$X = C \times (1 - |(H / 60^\circ) \bmod 2 - 1|)$$

$$m = V - C$$

$$(R', G', B') = \begin{cases} (C, X, 0) & , 0^\circ \le H < 60^\circ \\ (X, C, 0) & , 60^\circ \le H < 120^\circ \\ (0, C, X) & , 120^\circ \le H < 180^\circ \\ (0, X, C) & , 180^\circ \le H < 240^\circ \\ (X, 0, C) & , 240^\circ \le H < 300^\circ \\ (C, 0, X) & , 300^\circ \le H < 360^\circ \end{cases}$$

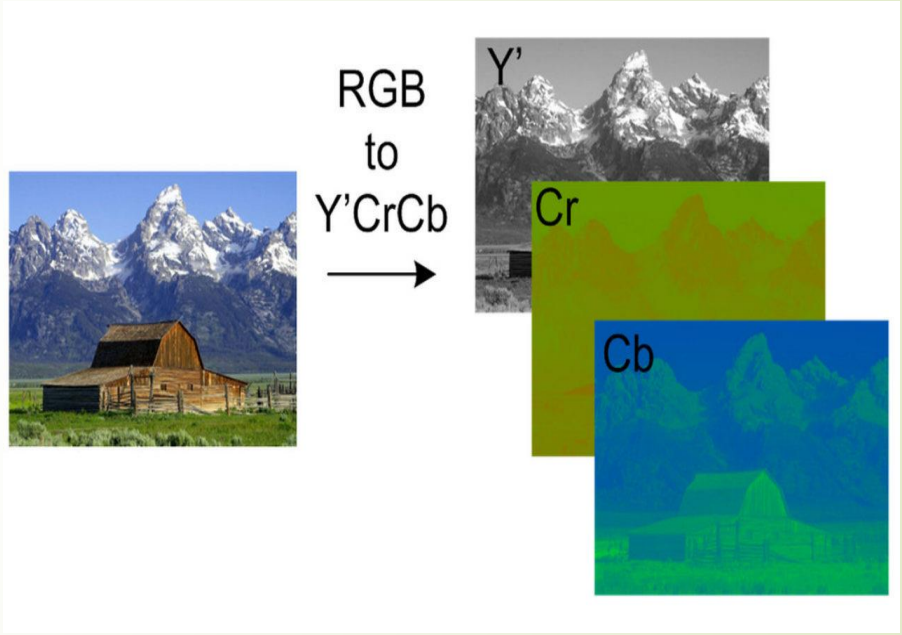
$$(R,G,B) = ((R'+m)\times 255, (G'+m)\times 255, (B'+m)\times 255)$$

Color Spaces (YCbCr)

The YCbCr color space separate the image into three components:

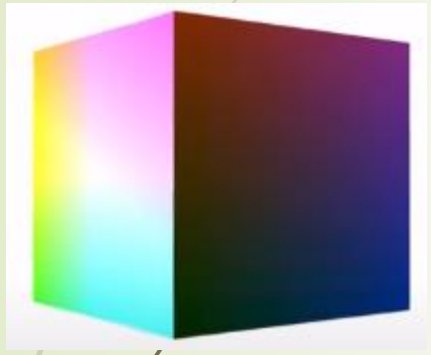
- ***The luminance*** component which represents the brightness (like gray-scale).
- ***Cb*** is the chrominance blue value
- ***Cr*** is the chrominance red value

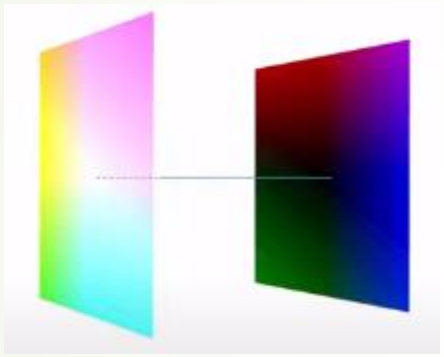
We can notice that luminance (achromatic) and color (chromatic) components are separated.



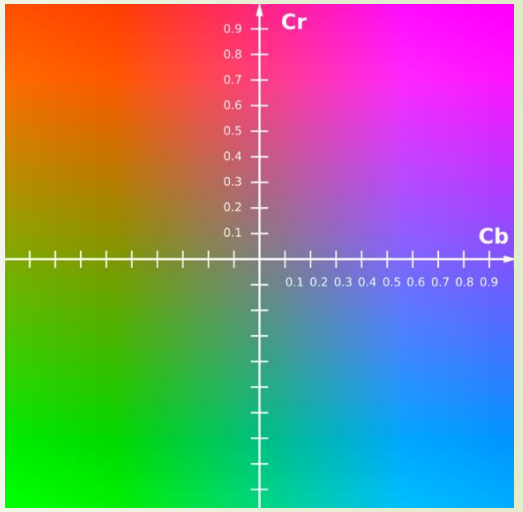
Color Spaces (YCbCr)

YCbCr can be visualized in 3 dimensions





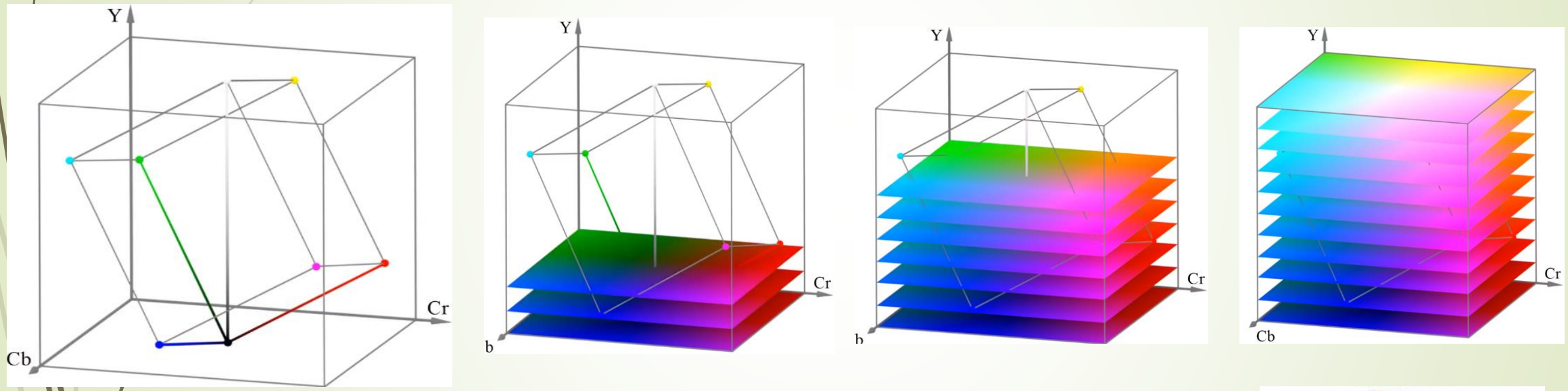
Y = 0.5



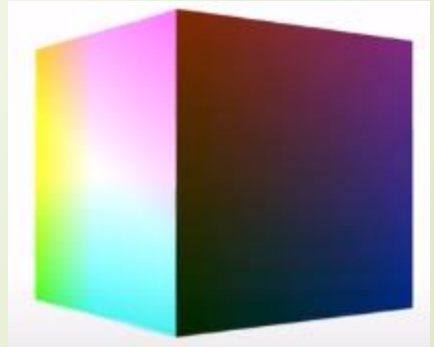
The center of the cube face is ***white***, and the center of the opposite side is ***black***. The line linking the both sides is the luminance component (from black to white). Therefore, each color can be represented by the center of one thin slice from the cube (luminance) + coordinates in the plane of ***Cb / Cr.***

Color Spaces (YCbCr)

The RGB color space inside the YCbCr, here is shown how cube slices are constructed

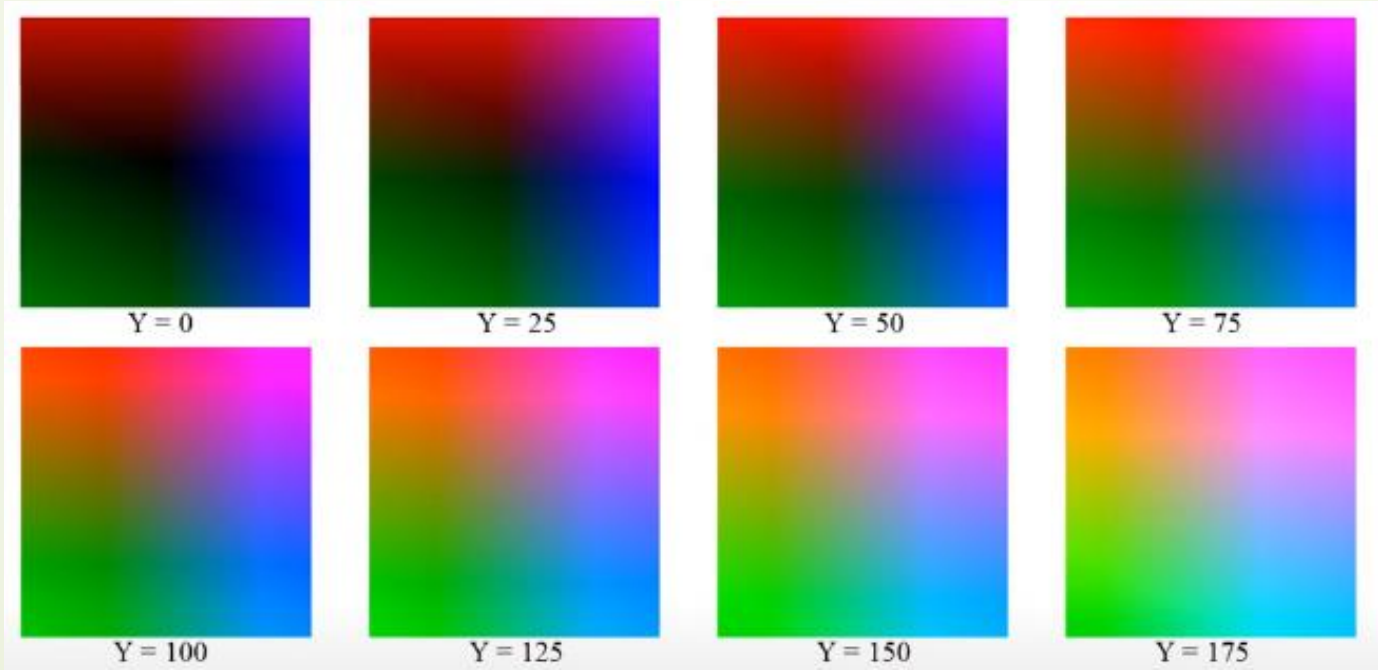


We can notice the change in CbCr planes with the intersection with the RGB cube



Color Spaces (YCbCr)

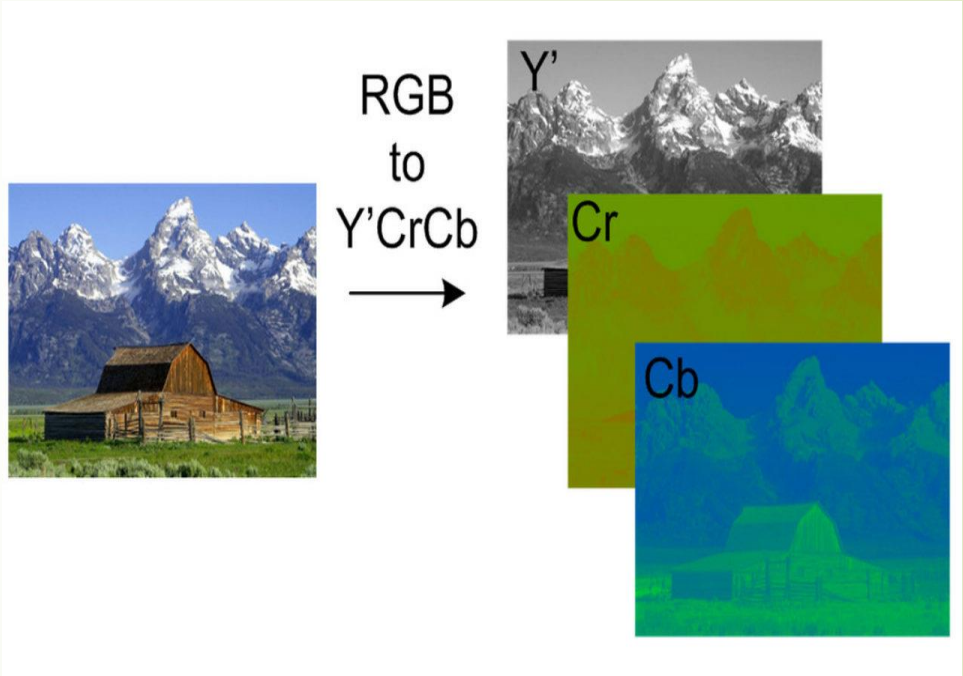
CbCr planes for different values of Y



Color Spaces (YCbCr)

A highly desirable property of YCbCr

Researchers have shown that humans are more sensitive to different levels of brightness than it is to differences in color. Since the brightness component (Y) is separated from the color (chromatic) components, these components are subsampled e.g., instead of considering 255 level (i.e., 8 bits), we can consider 16 level (i.e., 4 bits). This is called chroma. subsampling and it allows reducing the amount of bit allocated to the image.



Color Spaces (YCbCr)

Conversion between RGB and YCbCr

$$
\begin{aligned}
Y &= 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \\
Cb &= 0.564 \cdot (B - Y) + 128 \\
Cr &= 0.713 \cdot (R - Y) + 128
\end{aligned}
$$

$$
\begin{aligned}
R &= Y + 1.403 \cdot (Cr - 128) \\
G &= Y - 0.344 \cdot (Cb - 128) - 0.714 \cdot (Cr - 128) \\
B &= Y + 1.773 \cdot (Cb - 128)
\end{aligned}
$$

Image Basics

- **Quantization Algorithm**

Algorithm: Quantization;

Input: image **I (N,M)**; int **GL_Values** [1..K]; int **Nbre_Levels**; Boolean **B**;
Output: image **QI**;

```
Begin
For i=1 to N
  For j=1 to M
     B = Faux;
     For t=1 to K
        if (GL_Value[t] >= I(i,j) )
           QI(i,j) = GL_Value[t];
            B = Vrai;
            break;
        end
     end
If (B == Faux)
QI(i,j) = GL_Value[end];  % or 255
End end end
End.
```

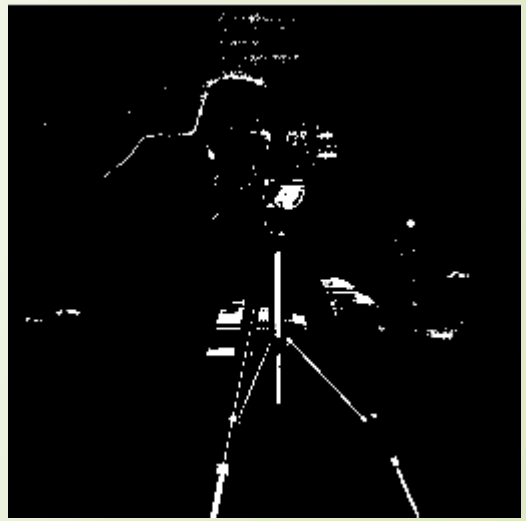
Image Basics

- **Morphological image processing**

We can perform a multitude of morphological operations on the binary and gray-scale images involving addition, complementary, logic operations…etc. Note that binarization process can be carried out according to a certain threshold.

Threshold = 125

Threshold = 50

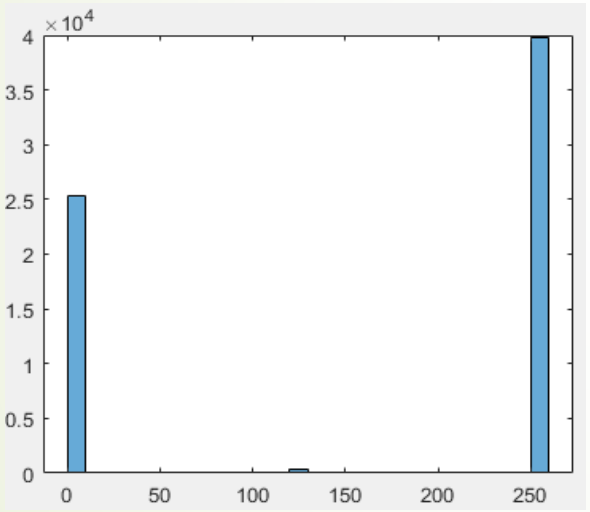


Threshold = 190

Image Basics

- **Morphological image processing**

Complementary image: replace the values of 0 by 255 and 255 to 0 in the binary images.



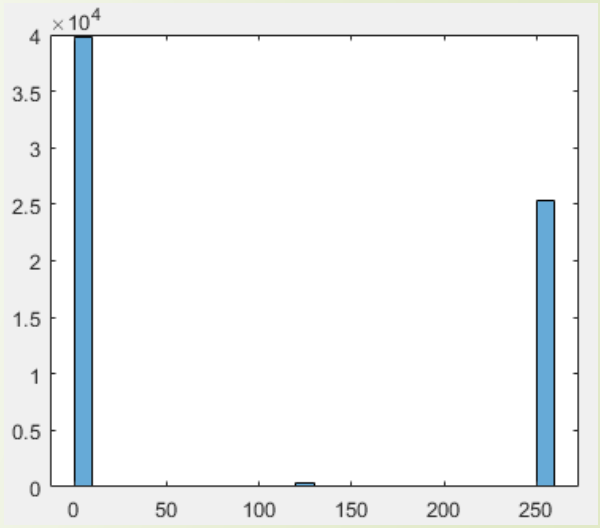
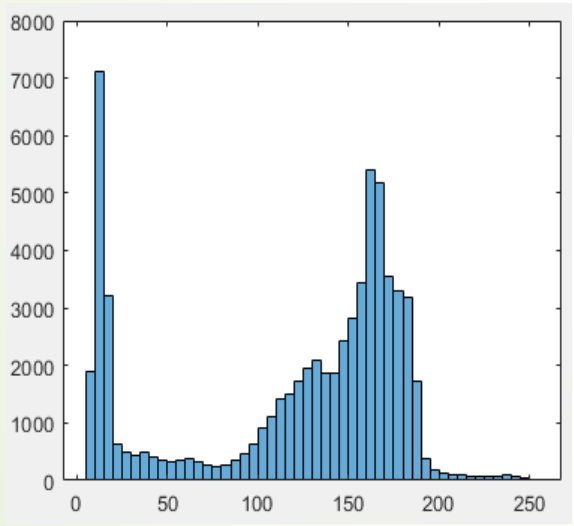


Image Basics

- **Morphological image processing**

Complementary image: the concept of complementarity can be generalized to gray-scale images where each value is replaced by its complementary to 255 i.e., X = 255 - X



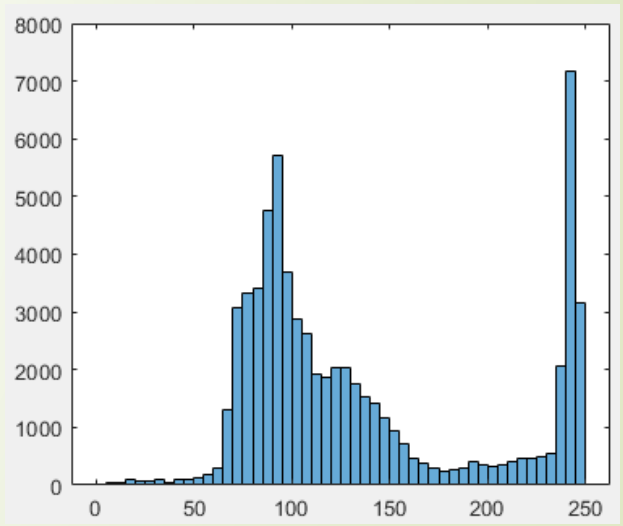
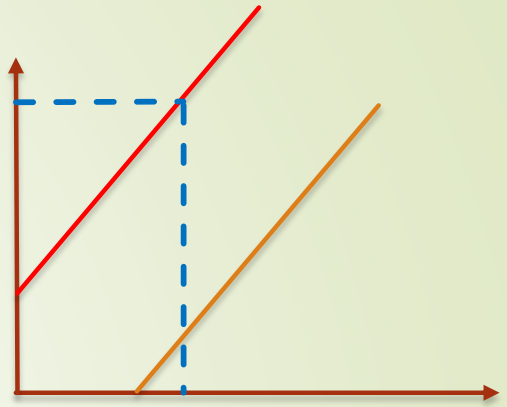
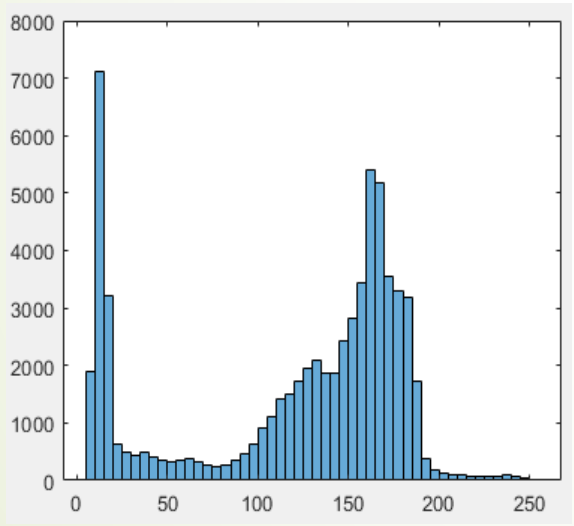


Image Basics

- **Morphological image processing**



Addition: which is equivalent to increasing image brightness X = X + A (A is the added value).
Subtraction has the inverse impact on the image.



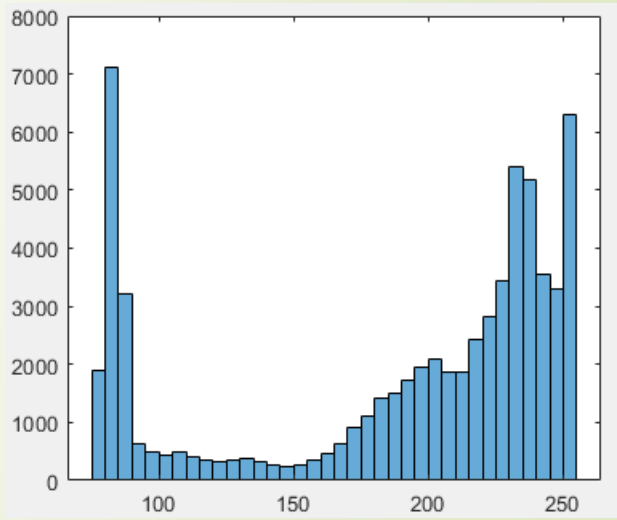
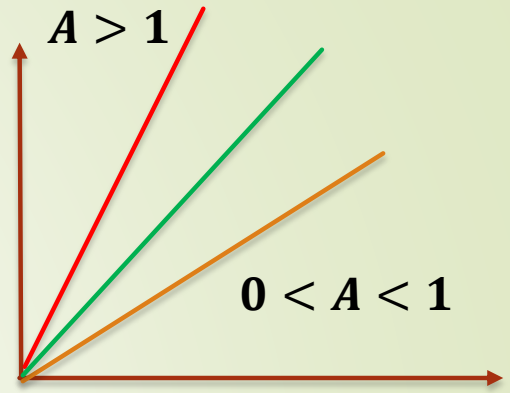
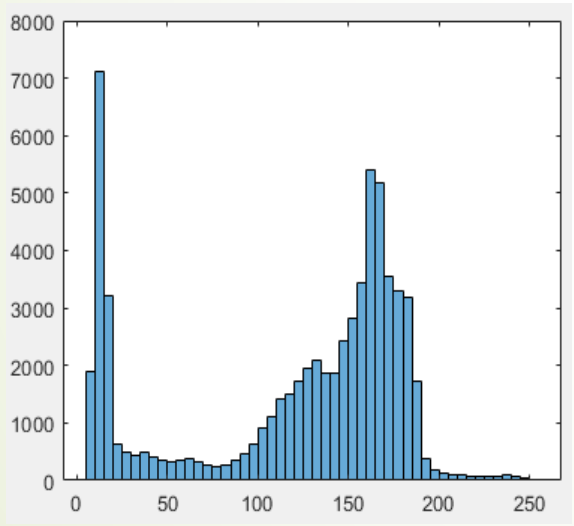


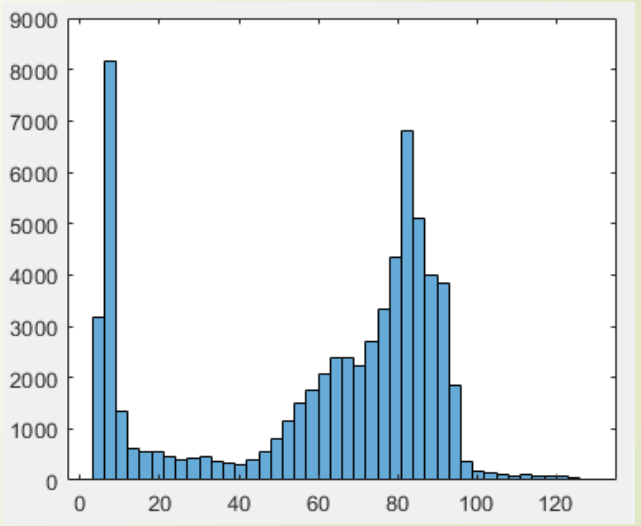
Image Basics

- **Morphological image processing**



Multiplication: which leads to increase / decrease the image brightness depending on the value By which the image values are multiplied $X = X \times A$. If $0 < A < 1$ then brightness will be Decreased, if $A > 1$ brightness will increase.

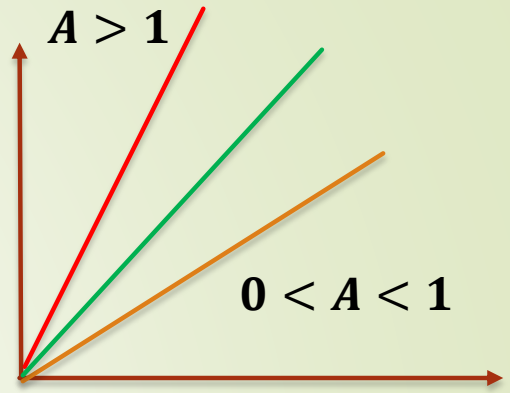




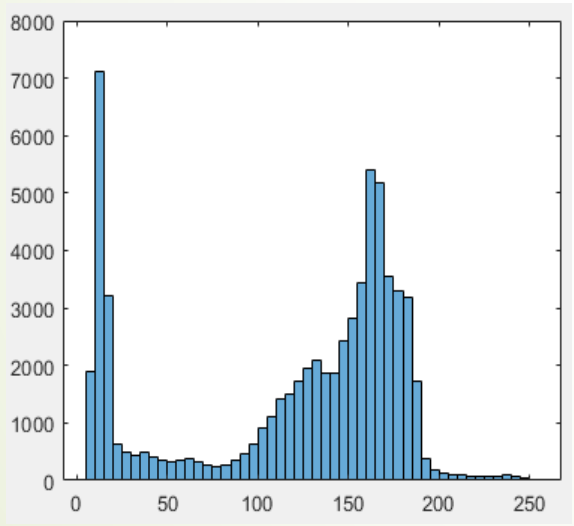
$A = 0.5$

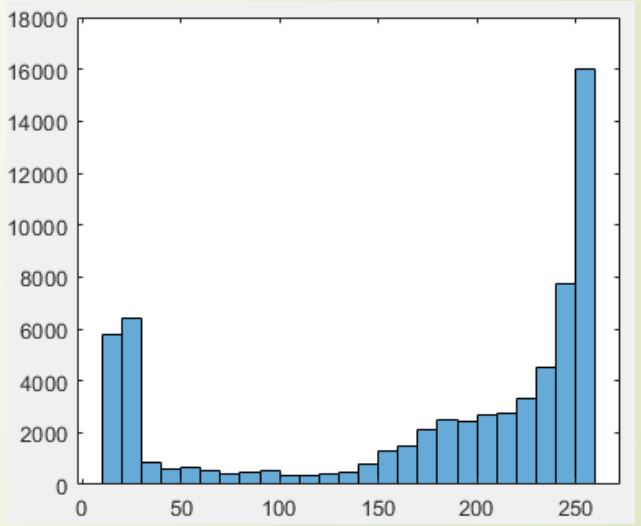
Image Basics

- **Morphological image processing**



Multiplication: which leads to increase / decrease the image brightness depending on the value By which the image values are multiplied $X = X \times A$. If $0 < A < 1$ then brightness will be Decreased, if $A > 1$ brightness will increase.





$A = 1.5$

Image Basics

- **Morphological image processing**

Division: $X = \frac{X}{A}$. If $0 < A < 1$ then brightness will be increased, if $A > 1$ brightness will decrease.

$\boldsymbol{A = 0.5}$

$\boldsymbol{A = 2}$

Image Basics

- **Morphological image processing**

Max/ Min: takes the min / max from two different images.

MAX

MIN

Image Basics

- **Morphological image processing**

Linear combination: can be done by the following equation $Z = aX + bY$

$\boldsymbol{a = 0.9, b = 0.1}$

$\boldsymbol{a = 0.4, b = 0.6}$

$\boldsymbol{a = 0.1, b = 0.9}$

Image Basics

- **Morphological image processing**

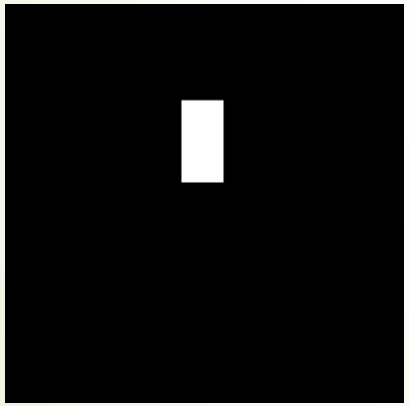
Logic operations: the previous operations were arithmetique operations, the logic operations include AND, OR, XOR, XNOR and other logic functions.

A	B	A AND B	A OR B	A XOR B	A XNOR B
0	0	0	0	0	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	1	0	1

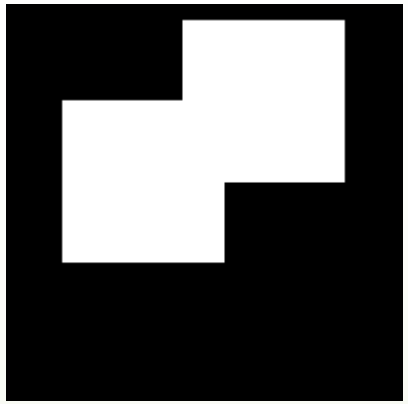
Image Basics

- **Morphological image processing**

Logic operations: the previous operations were arithmetique operations, the logic operations include AND, OR, XOR, XNOR and other logic functions.

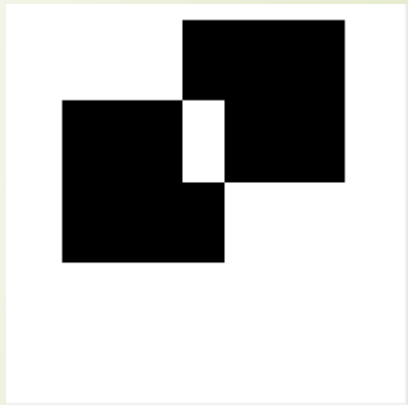


A AND B



A OR B

A XOR B

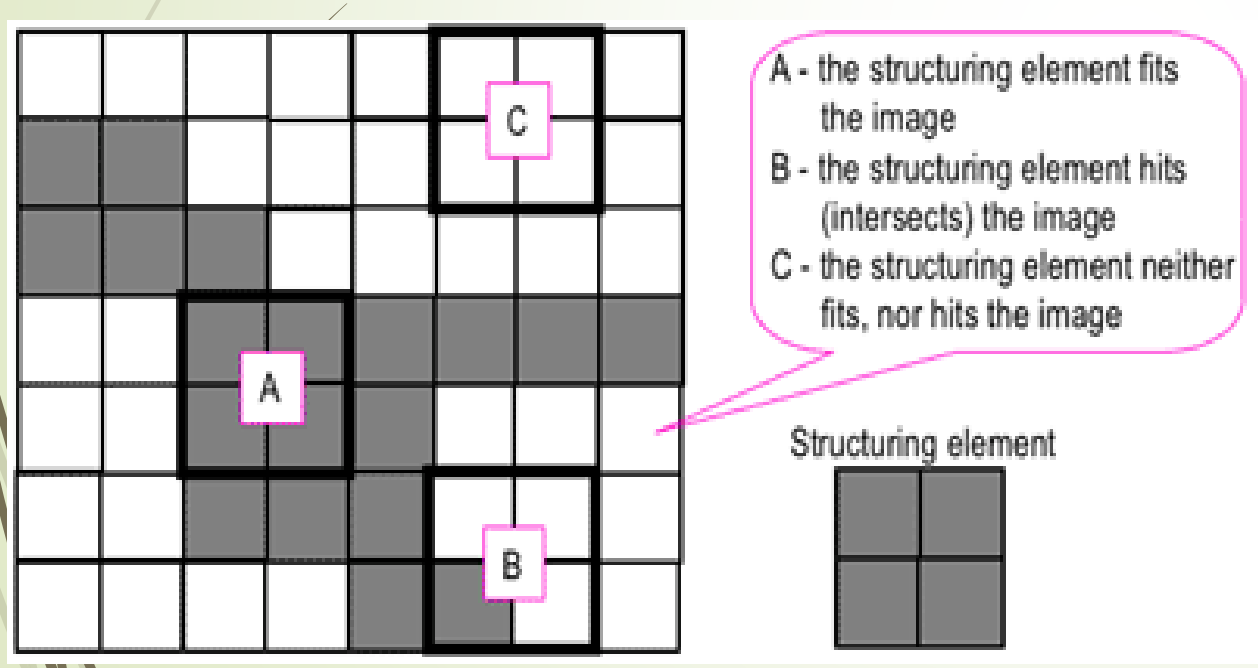


A XNOR B

Image Basics

- **Morphological image processing**

Erosion and dilation: these two operations are based on the concept of the structural element (small binary image). This element browse the image by passing through all the possible locations in the image, and it is compared with the corresponding neighborhood of pixels.



The structural element could take other forms e.g.,

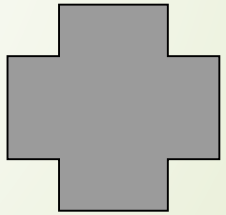
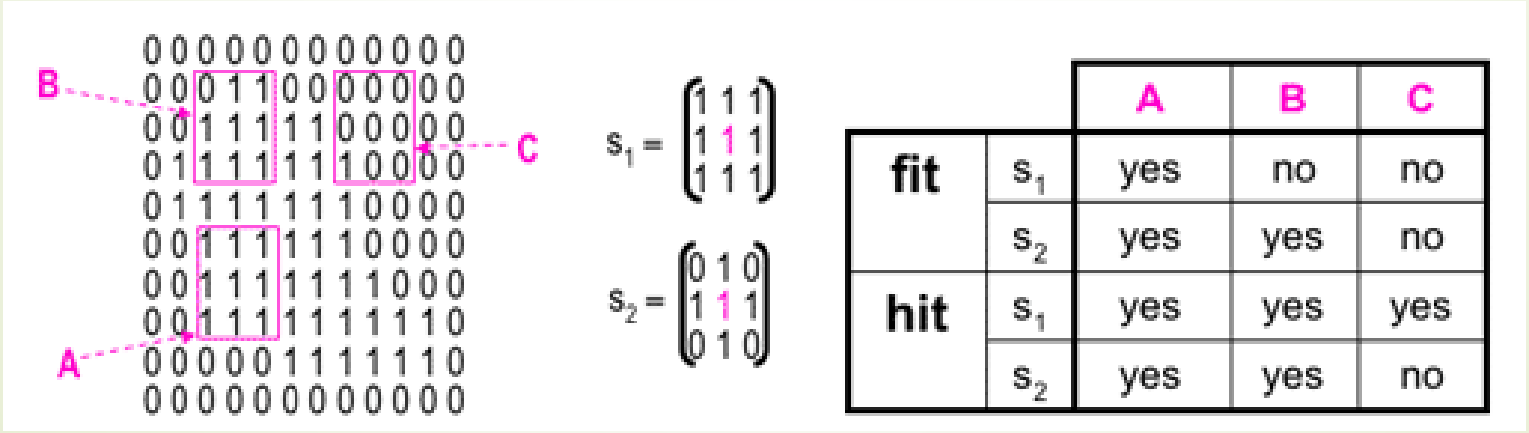


Image Basics

- **Morphological image processing**

Erosion and dilation:

The structuring element is said to ***fit*** the image if, for each of its pixels set to 1, the corresponding image pixel is also 1. Similarly, a structuring element is said to ***hit***, or intersect, an image if, at least for one of its pixels set to 1 the corresponding image pixel is also 1.

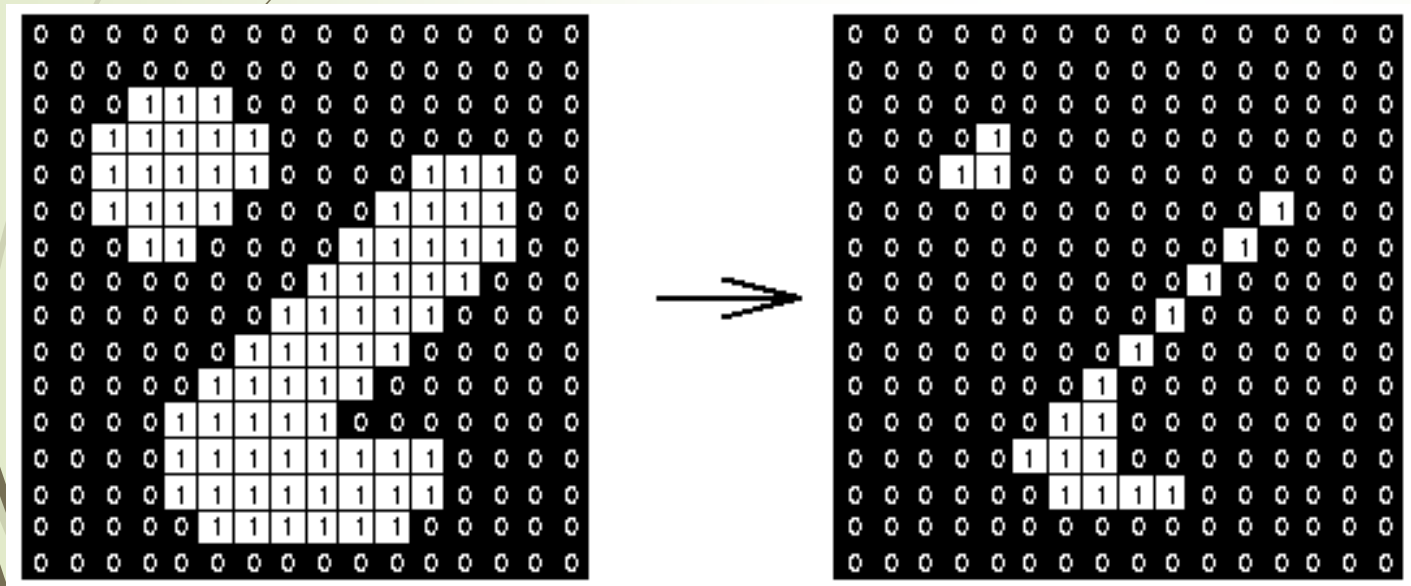


		A	B	C
fit	s_1	yes	no	no
	s_2	yes	yes	no
hit	s_1	yes	yes	yes
	s_2	yes	yes	no

Image Basics

- **Morphological image processing**

Erosion of a binary image ***f*** with a structural element ***s*** denoted $f \ominus s$ yields a binary image $g = f \ominus s$ with ones in all locations (x, y) of structuring element's origin at which that element fits the input image i.e., $g(x, y) = 1$ is ***s*** fits ***f*** and **0** otherwise, repeating for all pixel coordinates (x, y).



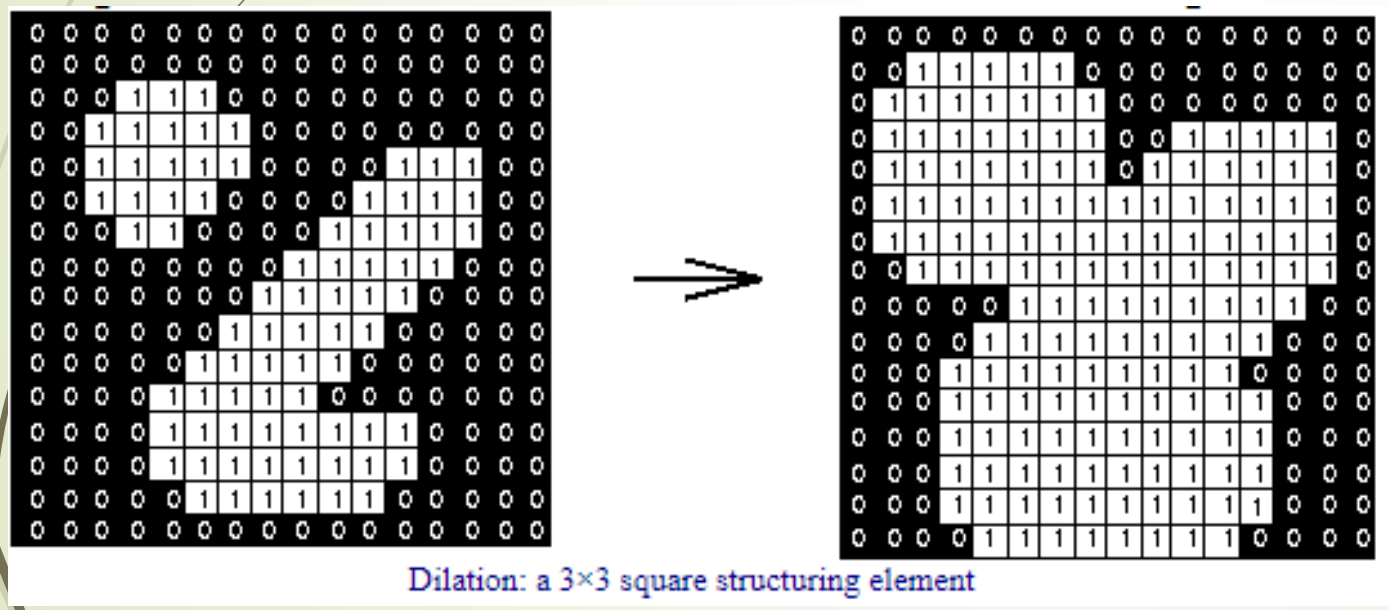
Erosion: a 3×3 square structuring element

square structuring elements shrinks an image by stripping away a layer of pixels from both the inner and outer boundaries of regions. The holes and gaps between different regions become larger, and small details are eliminated. The boundaries can be recovered by $b = f - (f \ominus s)$

Image Basics

- **Morphological image processing**

Dilation of a binary image $\boldsymbol{f}$ with a structural element $\boldsymbol{s}$ denoted $f \oplus s$ yields a binary image $g = f \oplus s$ with ones in all locations (x, y) of structuring element's origin at which that element hits the input image i.e., $g(x, y) = 1$ is $\boldsymbol{s}$ hits $\boldsymbol{f}$ and **0** otherwise, repeating for all pixel coordinates (x, y).



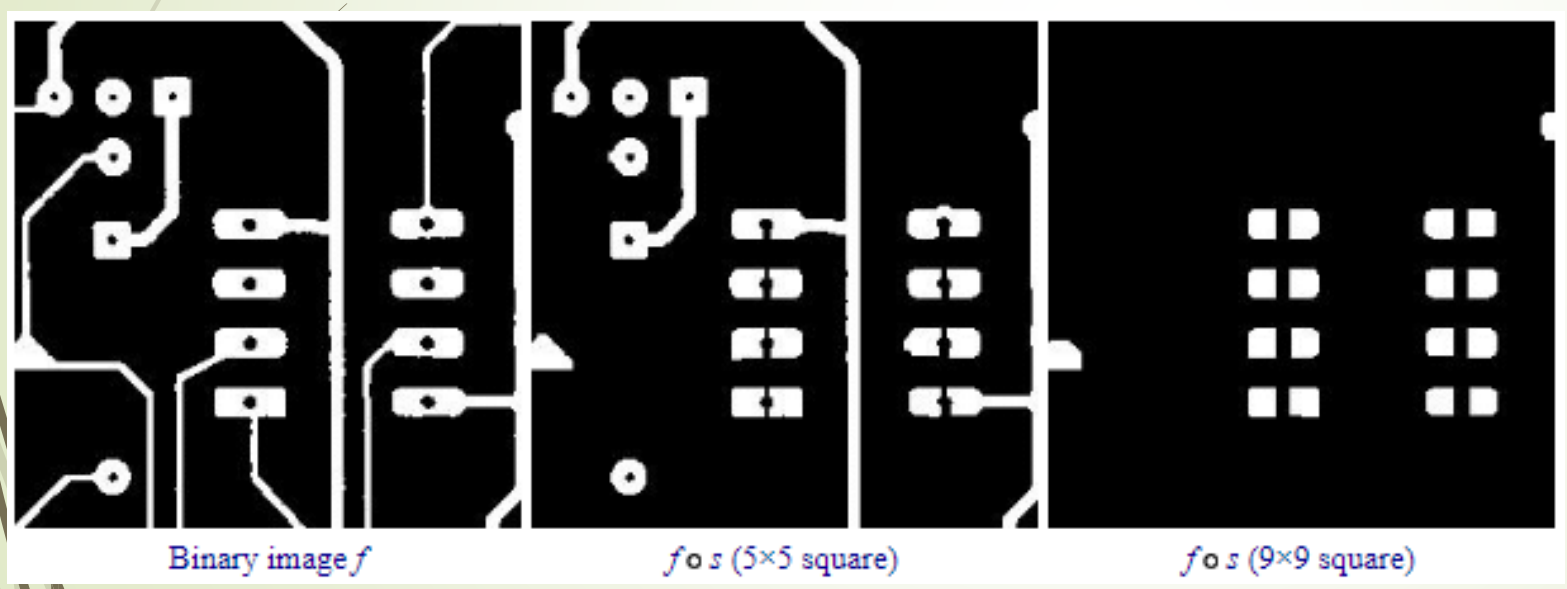
Dilation: a 3×3 square structuring element

The holes enclosed by a single region and gaps between different regions become smaller, and small intrusions into boundaries of a region are filled in

Image Basics

- **Morphological image processing**

Opening of a binary image ***f*** with a structural element ***s*** denoted $f \circ s$ is an erosion followed by dilation $f \circ s = (f \ominus s) \oplus s$

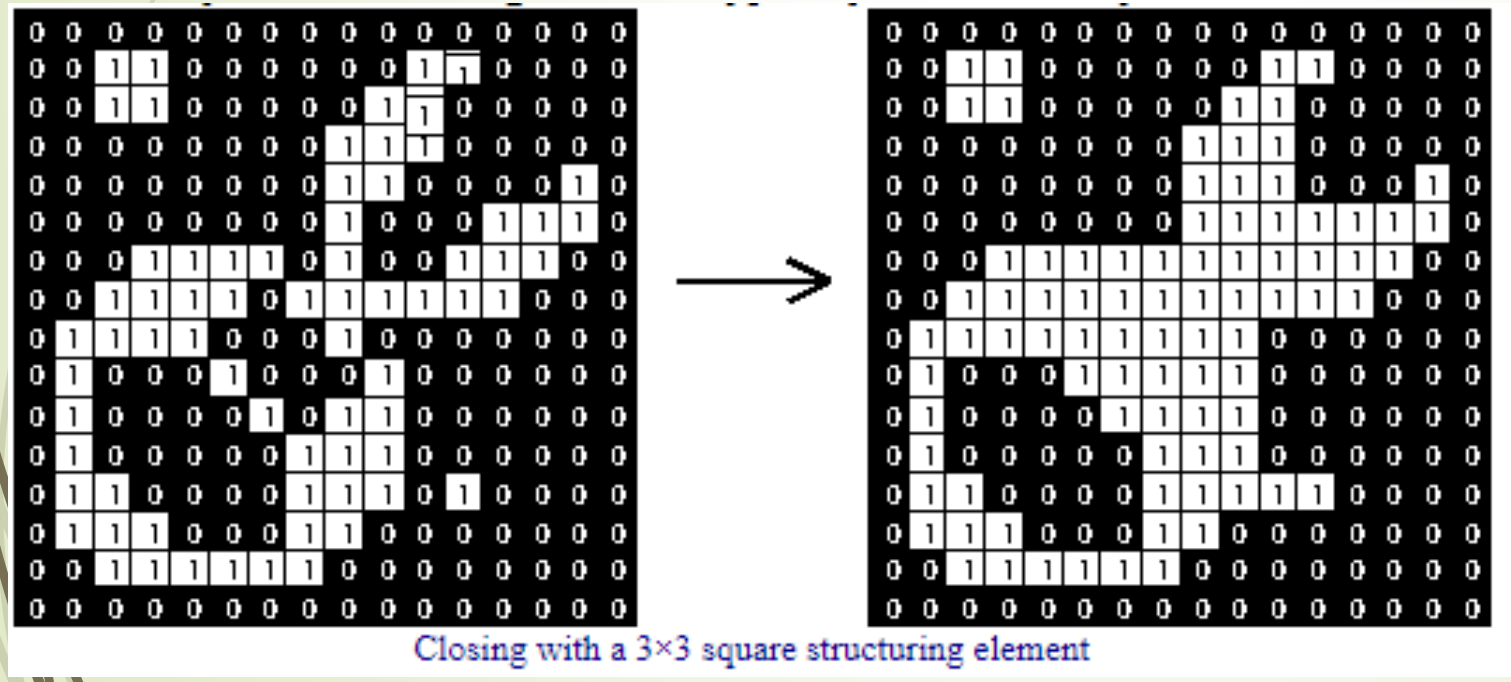


Opening is so called because it can open up a gap between objects connected by a thin bridge of pixels.

Image Basics

- **Morphological image processing**

Closing of a binary image ***f*** with a structural element ***s*** denoted $f \cdot s$ is a dilation followed by erosion $f \cdot s = (f \oplus s) \ominus s$

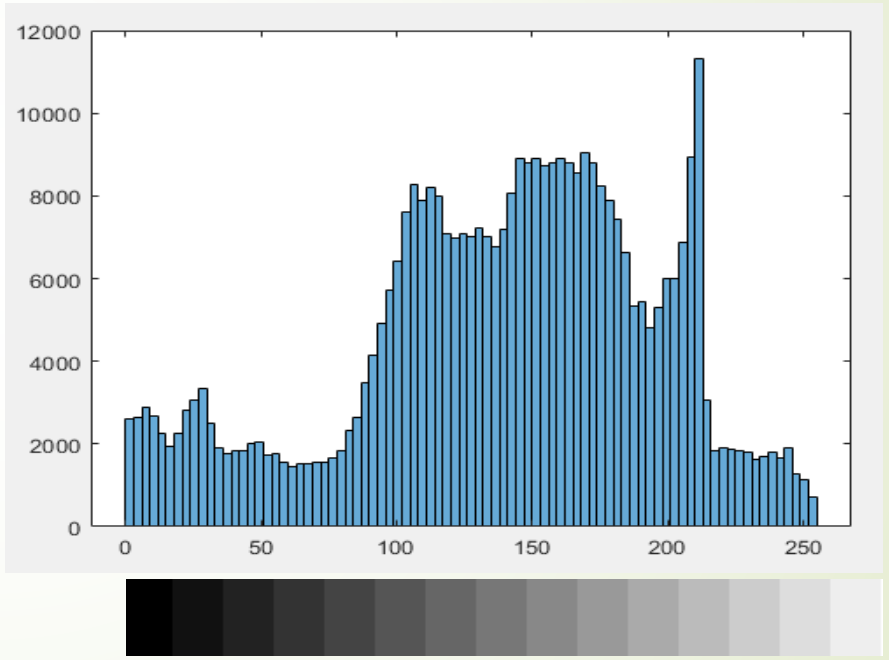


Closing with a 3×3 square structuring element

Closing is so called because it can fill holes in the regions while keeping the initial region sizes.

Improving Image quality

- **Gray-level image histogram**



On the y axis of this histogram are the frequency or count. And on the x axis, we have gray level values.

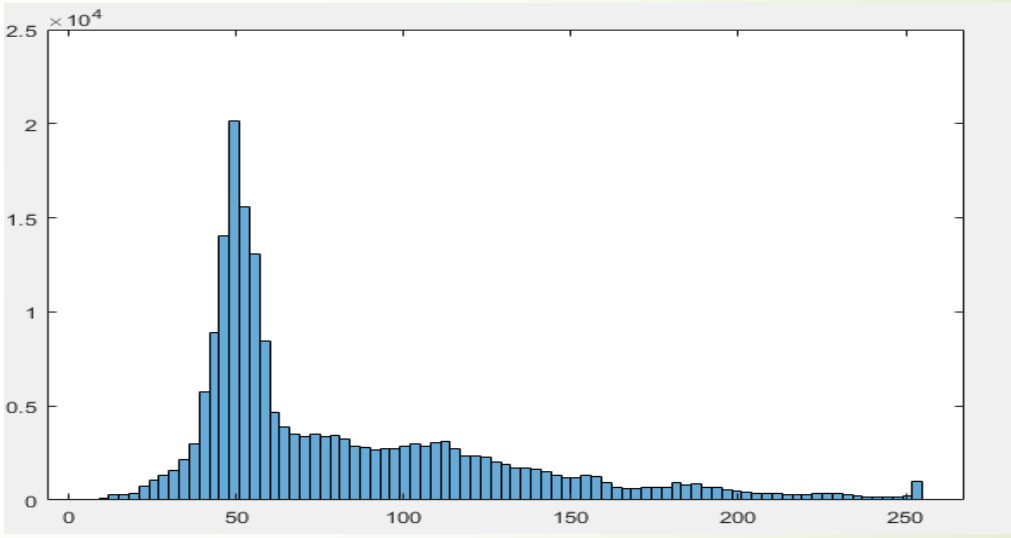
Improving Image quality

- **Gray-level image histogram**

However, one shortcoming of histogram is that it doesn’t encode spatial information. In other words, we can use it for counting intensities frequency, but not their distribution.

Improving Image quality

- **Improving image contrast using histogram sliding**



We can notice that frequent intensities are in the first half of the histogram (< 60). Thus, image tend to be dark. To get a brighter one, we perform histogram sliding.

Improving Image quality

- **Increase Brightness based on histogram sliding**

Algorithm: Increase_Brightness;

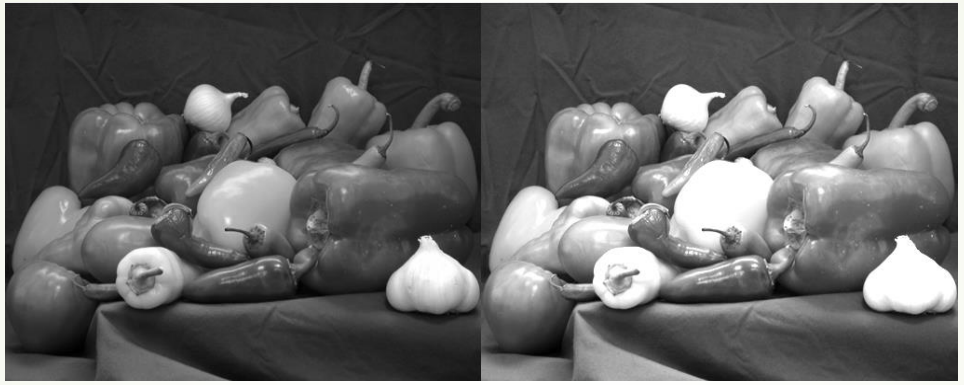
Input: image **I (N,M)**; int Value;
Output: image **I2**;
Begin

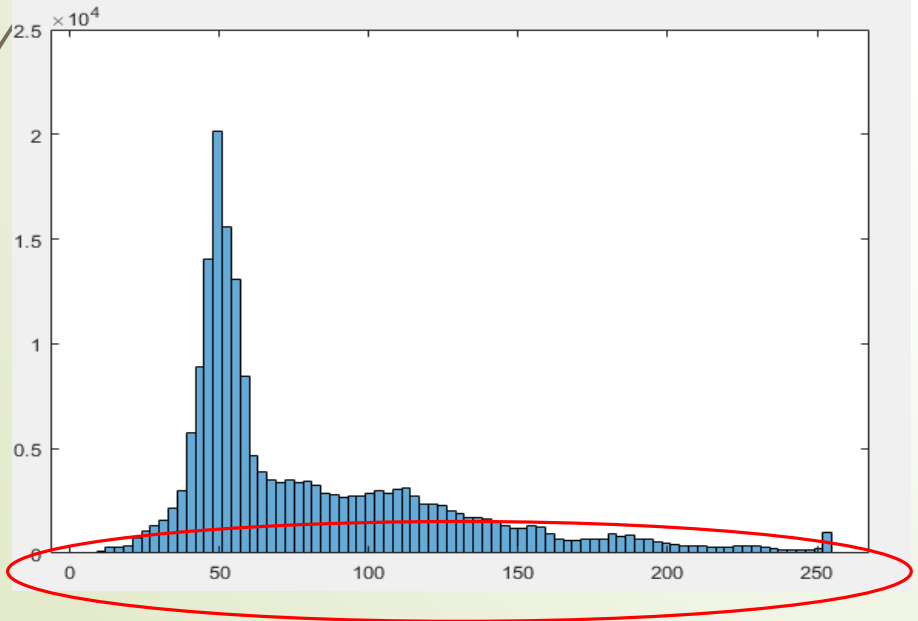
```
For i=1 to N
  For j=1 to M
        I2(i,j) = I(N,M) + Value;
  end
end
```

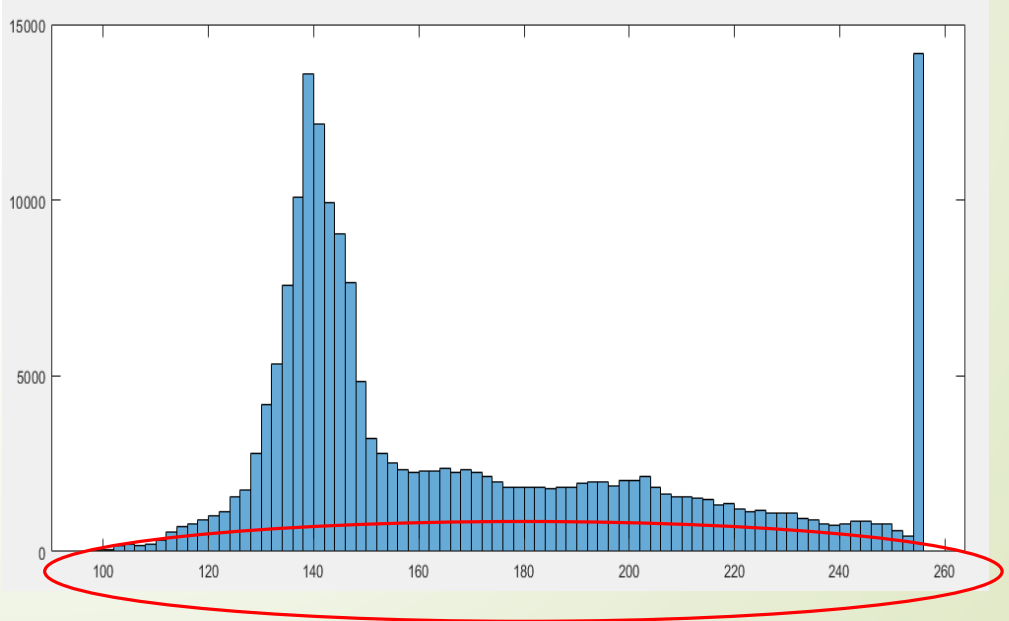
End.

Improving Image quality

- **Increase Brightness based on histogram sliding**



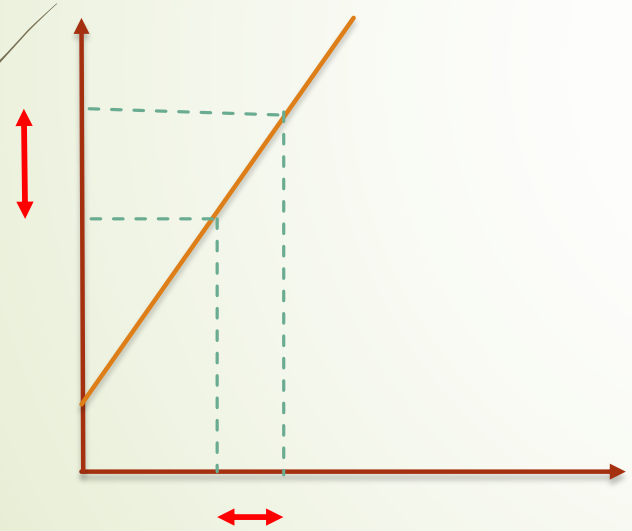




Improving Image quality

- **Adjust image dynamic to improve contrast (histogram stretching)**

Why $\boldsymbol{aX + b}$



$\boldsymbol{a}$ is the slope and represents how steep the line is

We should select the $\boldsymbol{a}$ that results in a bigger interval

$\boldsymbol{b}$ is the y-intercept

Improving Image quality

- **Adjust image dynamic to improve contrast (histogram stretching)**

The max value that R(i,j) can have is (Respectively min value)

$\boldsymbol{R_{max}}$ = a * I_{max} + b;

$\boldsymbol{R_{min}}$ = a * I_{min} + b;

We can solve this equations system using Cramer's rule. We find

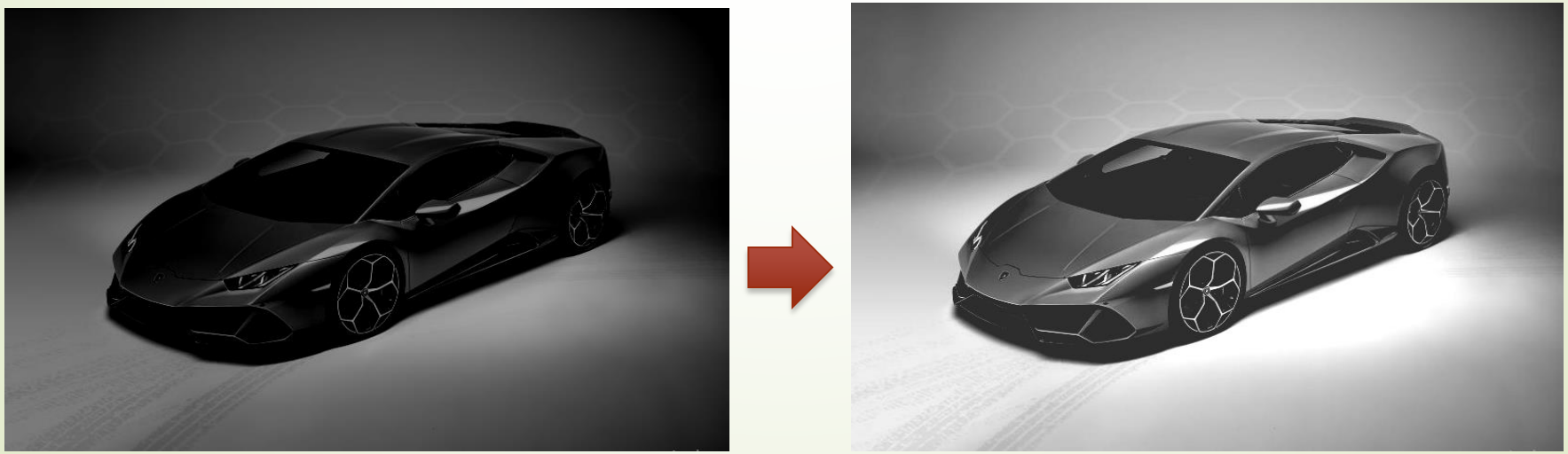
$$\boldsymbol{a} = \frac{\boldsymbol{R_{max} - R_{min}}}{\boldsymbol{I_{max} - I_{min}}} \qquad \boldsymbol{b} = \frac{\boldsymbol{I_{max} R_{min} - I_{min} R_{max}}}{\boldsymbol{R_{max} - R_{min}}}$$

Improving Image quality

- **Adjust image dynamic to improve contrast (histogram stretching)**

$$[I_{min}\ \ I_{max}] = [0\ \ 185] \longrightarrow [R_{min}\ \ R_{max}] = [50\ \ 255]$$

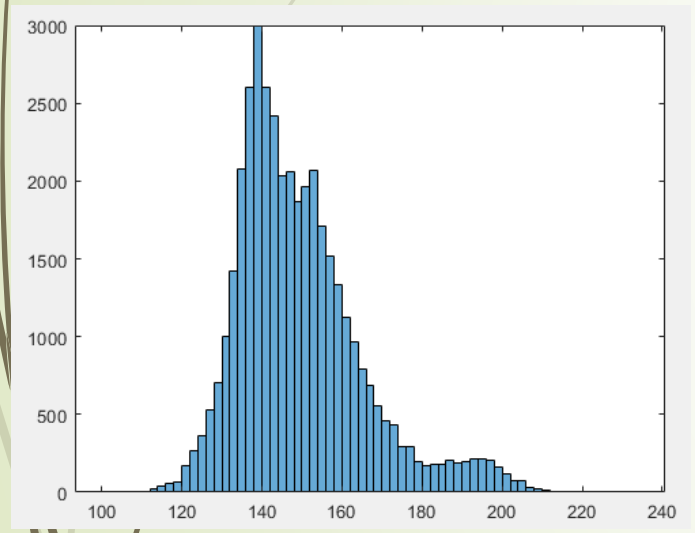
$$\boldsymbol{a = \frac{255 - 50}{185}} \qquad \boldsymbol{b = \frac{185 \times 50 - 0 \times 255}{255\ - 50}}$$

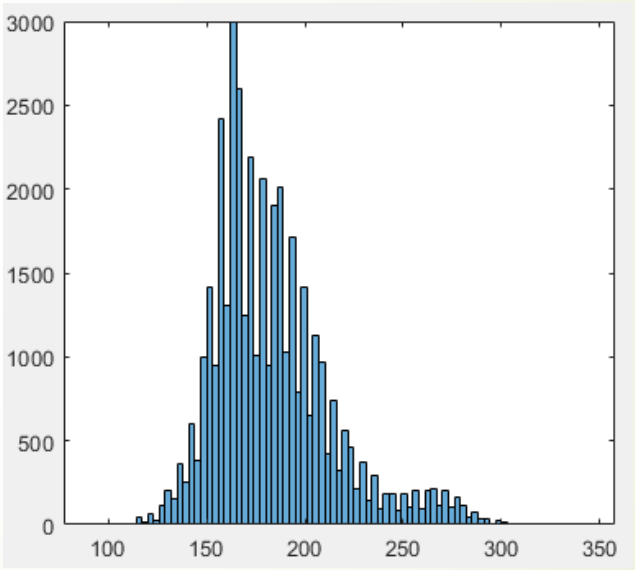


Improving Image quality

- **Adjust image dynamic to improve contrast (histogram stretching)**

$[I_{min}\ \ I_{max}] = [100\ \ 234] \longrightarrow [R_{min}\ \ R_{max}] = [0\ \ 255]$





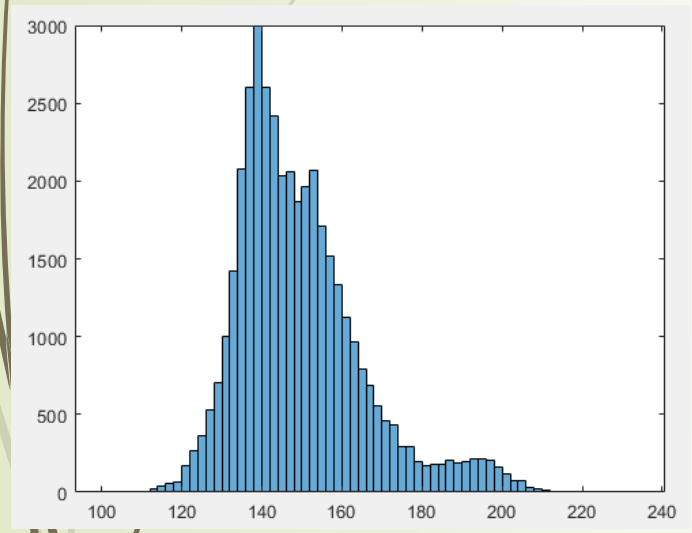
We can notice that the peaks are maintained

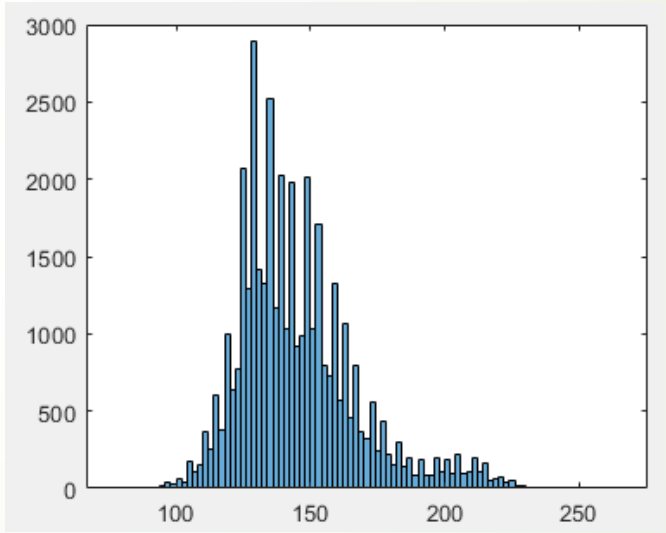
$$a = \frac{255-0}{234-100} = 1.9 \qquad b = \frac{234\times0-100\times255}{255-0} = -100$$

Improving Image quality

- **Adjust image dynamic to improve contrast (histogram stretching)**

$[I_{min}\ \ I_{max}] = [100\ \ 234] \longrightarrow [R_{min}\ \ R_{max}] = [50\ \ 240]$





We can notice that the peaks are maintained $\boldsymbol{a = \frac{240-50}{234-100} = 1.41}$ $\boldsymbol{b = \frac{234\times50-100\times240}{240-50} = -64.73}$

Improving Image quality

- **Adjust image dynamic to improve contrast (histogram stretching)**

$$\boldsymbol{Dynamic\ range} = Max\ intensity\ - Min\ intensity$$

Algorithm: Adjust;

Input: image **I (N,M)**; % in the range $[I_{min}\ I_{max}]$
Output: image **R**; % in the range $[R_{min}\ R_{max}]$ int $\boldsymbol{R_{min}\ R_{max}}$;
Begin

```
For i=1 to N
  For j=1 to M
     R(i,j) = a * I(i,j) + b;
   end
end
```

End.

Improving Image quality

- **Adjust image dynamic to improve contrast (histogram stretching)**

Another transformation can be expressed as

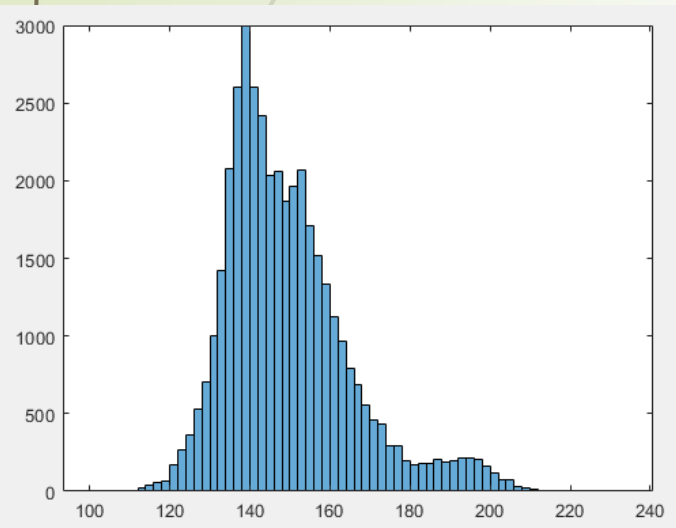
$$\boldsymbol{New = R_{max} \times \frac{X - I_{min}}{I_{max} - I_{min}}}$$

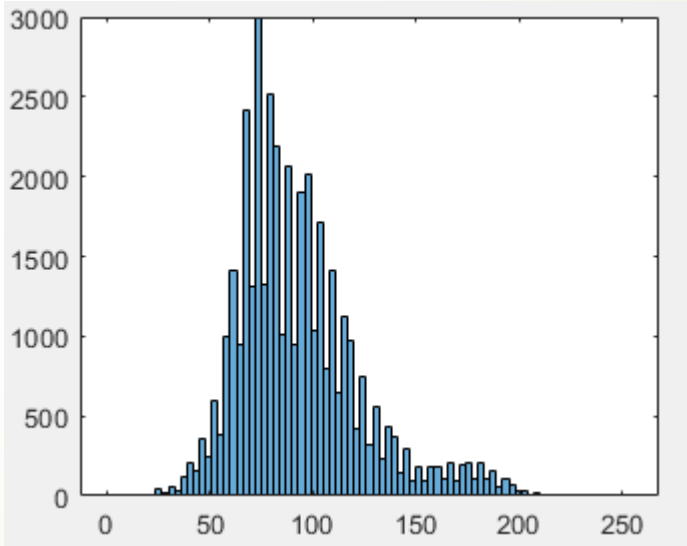
The second term will be between 0 and 1. and $\boldsymbol{R_{max}}$ is the highest value of output image.

Improving Image quality

- **Adjust image dynamic to improve contrast (histogram stretching)**

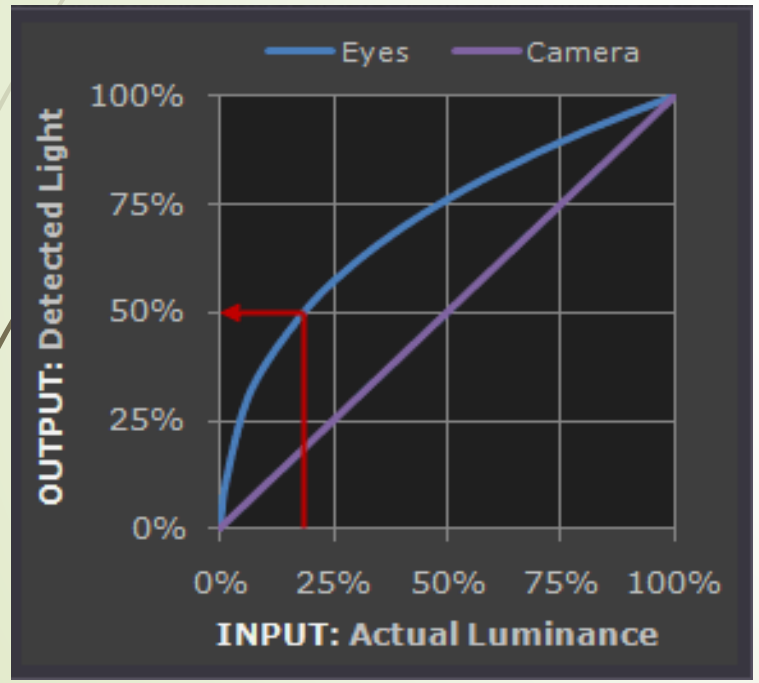
$$R_{max} = 255$$





Improving Image quality

- **Adjust image dynamic to using Gamma correction**

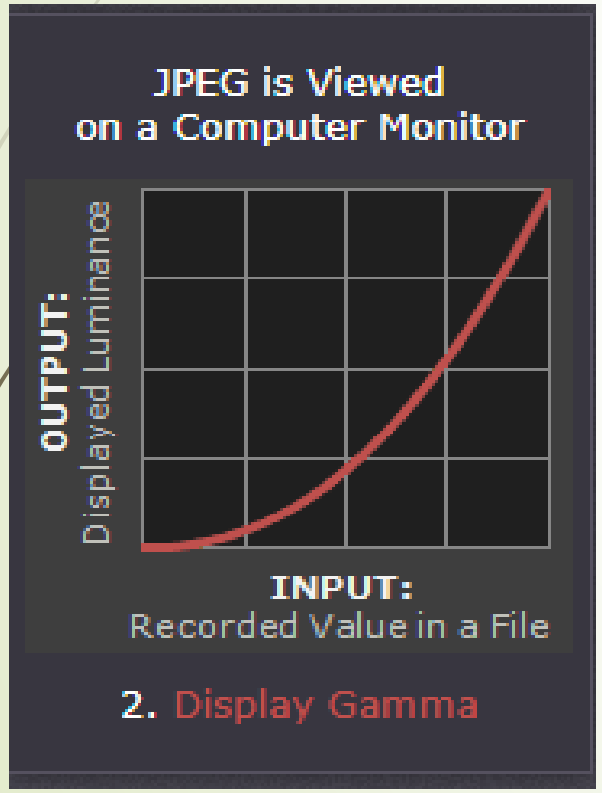


At first, we should know that our eyes don't perceive light as the camera does. We can see that we are much more sensitive to dark levels than bright ones. For instance, for an actual light (luminance) of about **20%**, it is perceived brighter (**50%**).

For the camera, a linear equation is considered $aX + b = 0$. However, it is not the case for the human eyes.

Improving Image quality

- **Adjust image dynamic to using Gamma correction**



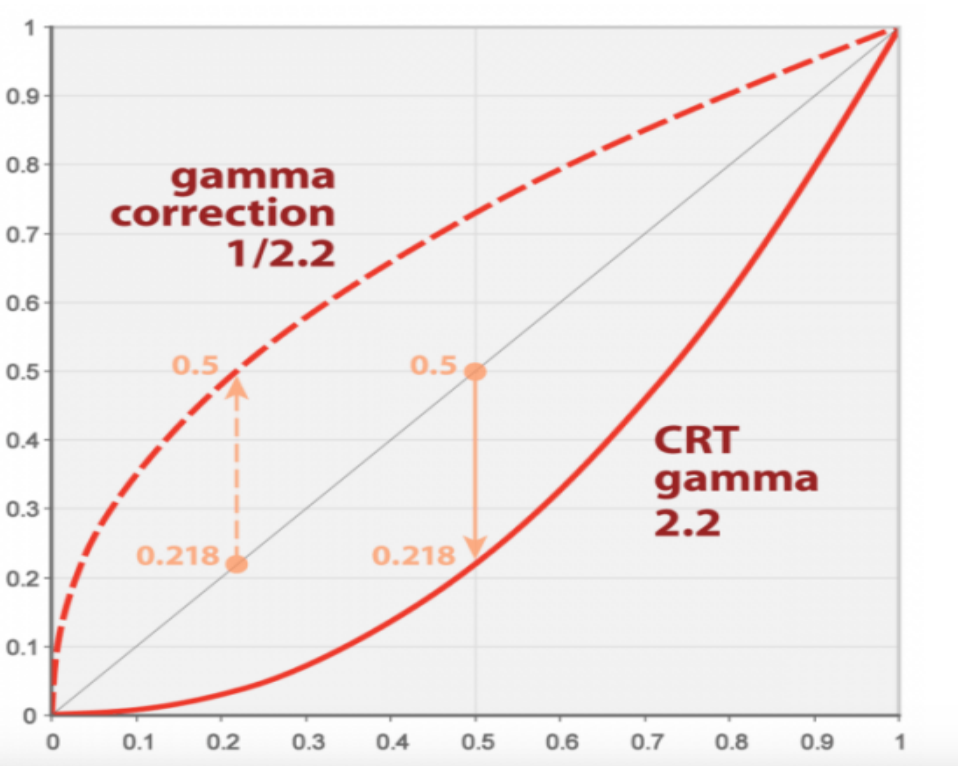
For a monitor, the light is viewed as in the figure (influence of video card and display device). The way the light is viewed by monitor can by expressed using the following equation

$$V_{out} = V_{in}^{\gamma}$$

Often, $\gamma = 2{,}2$ (note that values are represented between 0 and 1, e.g., $0.5^{2.2} = 0.21$

Improving Image quality

- **Adjust image dynamic to using Gamma correction**

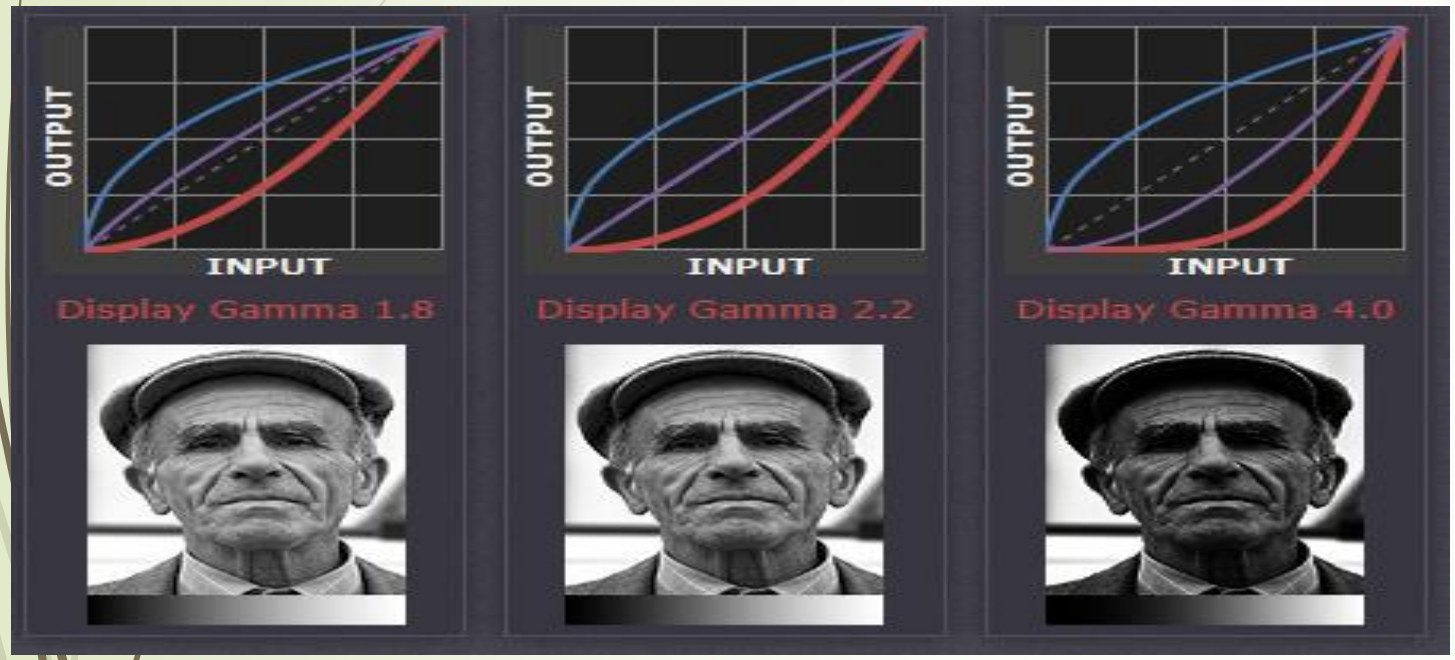


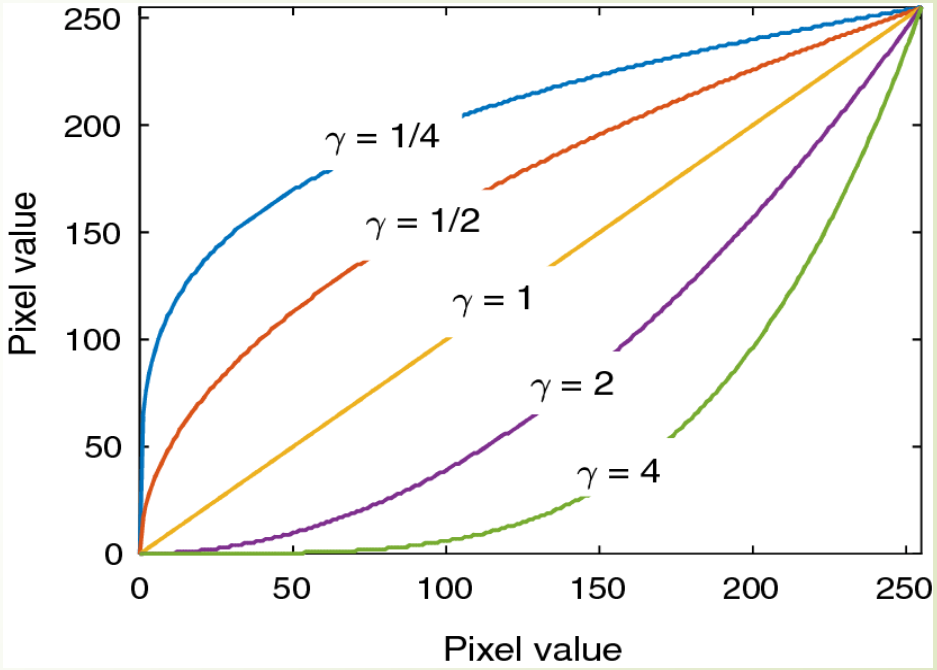
To overcome such an issue, Gamma correction is used. The camera stores the signal after performing Gamma correction, thus, original (**linear**) signal will be recovered and showed by monitors.

e.g., we have $0.5^{2.2} = 0.21$, to recover the original signal, we put $0.21^{1/2.2} \approx 0.5$

Improving Image quality

- **Adjust image dynamic to using Gamma correction**



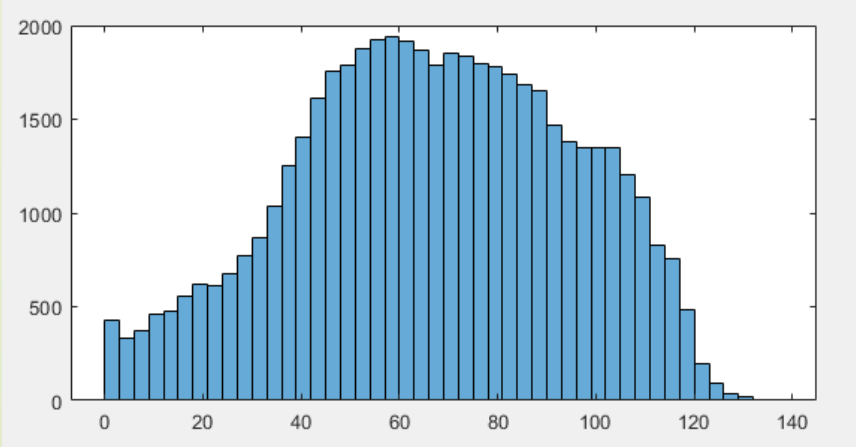


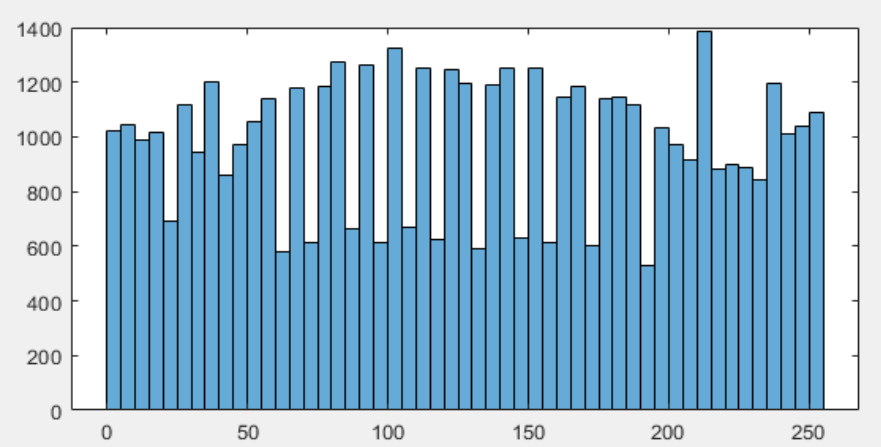
We notice that the more γ is higher (red curve), the more image is darkened (violet curve). (Blue curve is Gamma correction).

Improving Image quality

- **Adjust image dynamic (improve contrast) by histogram equalization**

The aim of this process is to increase the image dynamic, as shown by the two figures below (0~130 become 0~250)





Improving Image quality

- **Adjust image dynamic (improve contrast) by histogram equalization**

Intensity values in image 1 range from 0 to 120, whereas they range from 0 to 250 for image 2. For this reason, image 2 is more contrasted than image 1. To deal with that, we may perform a histogram equalization process (transformation) according to the following formula

$$N'_i = L_{max} \sum_{j=0}^{i} H(L_j) \qquad OR \qquad N'_i = L_{max} \times HC(L_j)$$

Such that H represents the PMF.
HC represents the CDF.
L_j is the current intensity of pixel.
L_{max} represents the maximum desired intensity
N'_i represents the new intensity value that replace L_j

Improving Image quality

- **Gray-level image histogram**

It count the occurrence frequency of each gray-level in the image.

$$H(i) = Number\ of\ pixels\ having\ i\ as\ intensity$$

Example

1	2	1	5
0	1	4	2
3	5	1	4
5	0	2	5

2	4	3	1	2	4

Improving Image quality

- **Gray-level image histogram**

Probability mass function (PMF)

1	2	1	5
0	1	4	2
3	5	1	4
5	0	2	5

2	4	3	1	2	4

2/16	4/16	3/16	1/16	2/16	4/16

Cumulative distribution function (CDF)

PMF	2/16	4/16	3/16	1/16	2/16	4/16
CDF	2/16	6/16	9/16	10/16	12/16	1

Improving Image quality

- **Adjust image dynamic (improve contrast) by histogram equalization**

Algorithm: Histeq;

Input: image **I (N,M)**; int L_{max};
Output: image **R(N,M)**; % improved image
Begin
$HC = CDF(I)$;
$Hist = Hist\ (I)$;
For i=1 to N
  For j=1 to M
    R(i,j) = $L_{max} \times HC(L_{i,j})$;
  end
end
End.

Improving Image quality

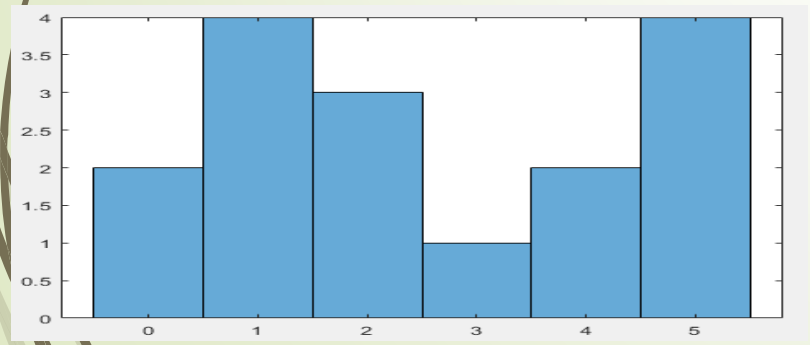
- **Example of histogram equalization**

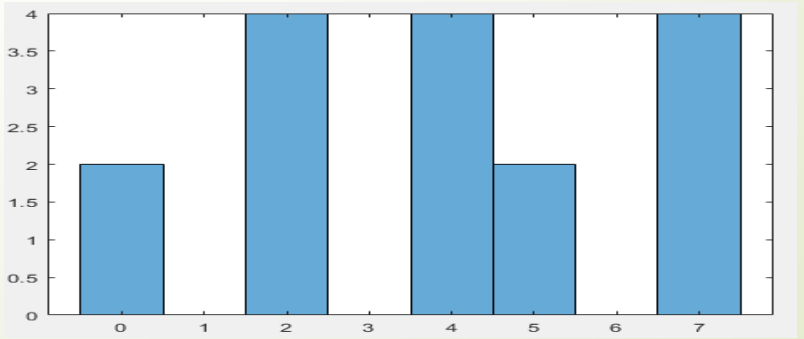
1	2	1	5
0	1	4	2
3	5	1	4
5	0	2	5

If the highest value in the image is 5, this means that this image can be encoded using three pixels. 3 bits encodes the range [0 7], thus, L_{max} in this case will equal 7 (the highest value in the range). Typically, L_{max} = 255, as images will be encoded using 8 bits. Here, the original gray levels are **[0 5],** and they are mapped to **[0 7].**

Intensity	0	1	2	3	4	5
N of pixels	2	4	3	1	2	4
PMF	0.066	0.26	0.2	0.066	0.13	0.26
CDF	0.066	0.326	0.526	0.592	0.722	1
CDF x L_{max}	0.46	2.24	3.64	4.13	5.04	7

2	4	2	7
0	2	5	4
4	7	2	5
7	0	4	7

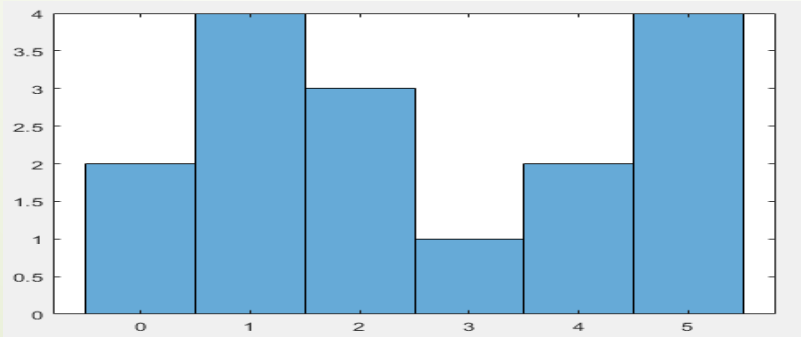


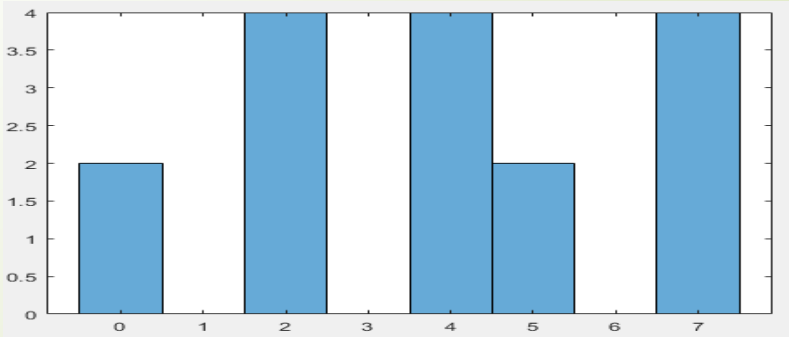


Improving Image quality

- **Adjust image dynamic (improve contrast) by histogram equalization**

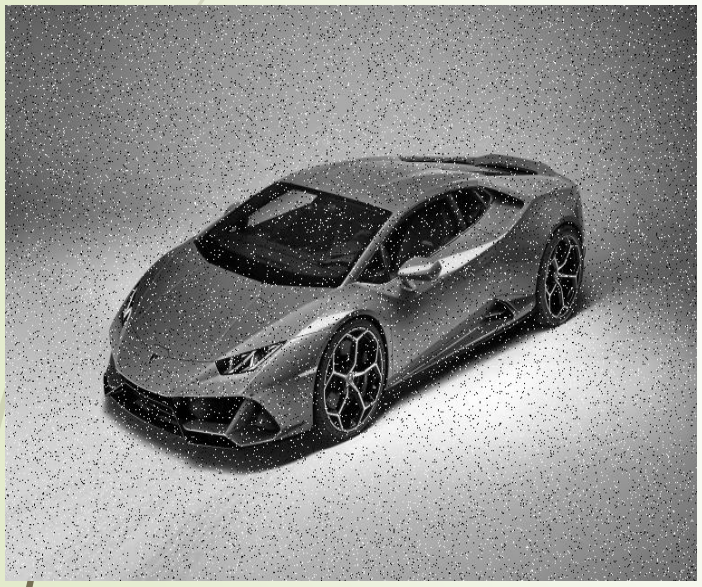
- The histogram equalization improve the contrast by mapping the gray-levels of the image to a new higher range e.g., **[0 5]** to **[0 7].**
- By this process the histogram will looks more flat and more evenly distributed (equalized).
- Also, the variance of the new range will typically be higher than the old range.
- The highest gray-level in the original range will be mapped to the highest value in the new range. This is simply because the CDF of the highest original value is 1 (then we will have $1 \times L_{max}$).
- The CDF of the highest original value is 1, and CDFs of the remaining values are proportional to this value (between 0 and 1), therefore, the values in the new range will be sorted ascending manner. We can notice that CDF is computed to transfer the values.
- The mapping to the new range will preserve the order of values in the original range as they are mapped using CDF.
- Histogram equalization does not maintain the peaks for two reasons. Firstly, because the range is stretched, thus, the peaks are dispersed e.g., peak of 1 is transfered to 2, and peak of 5 to 7. Second, because of the rounding effect two different values may be mapped to the same value e.g., 2 and 3 are mapped to 4.





Improving Image quality

- **Removing noise**



Mainly there are several reasons behind the presence of noise in the image.

- Environmental factors which can negatively influence the imaging sensor.
- Low light and sensor temperature may cause image noise.
- Problems of quantification.
- Dust in the scanner can cause noise.
- Interference during transmission.

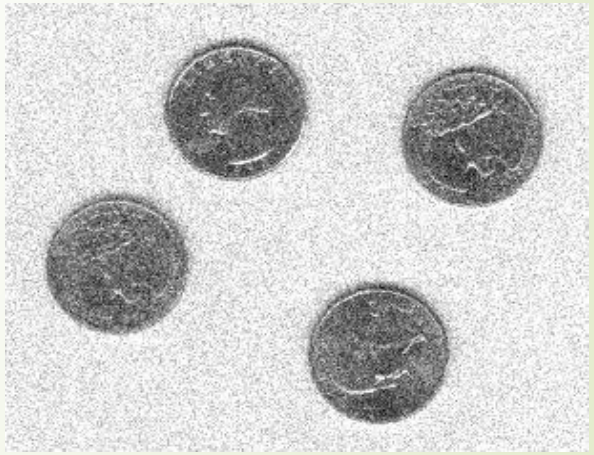
Improving Image quality

- **Removing noise**

There are several types of noise, among which we find the salt-pepper, and the Gaussian noise

Original image

Salt-pepper noise



Gaussian noise

Improving Image quality

- **Removing noise**

Salt and pepper noise

Salt Noise: it is added to an image by adding random bright point (value = 255) in all the image.

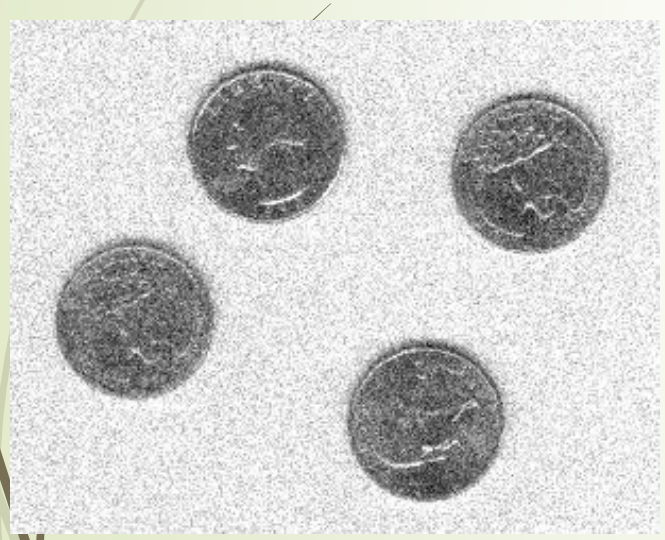
Pepper Noise: it is added to an image by adding random bright point (value = 0) in all the image.

Salt and Pepper Noise: as its name indicates this type of noise is added by considering both salt and pepper noises.

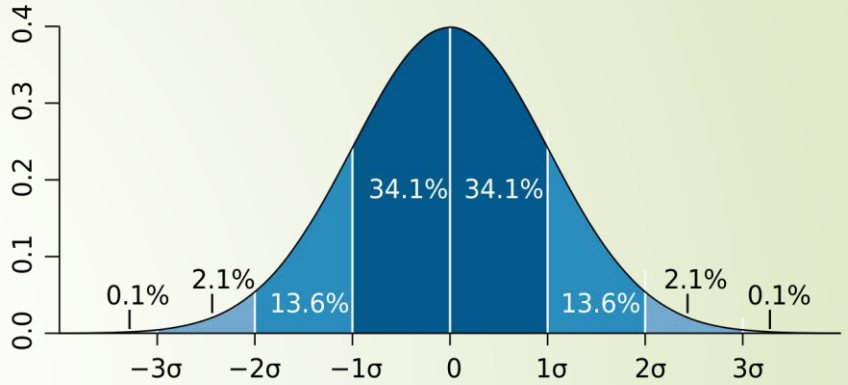
Improving Image quality

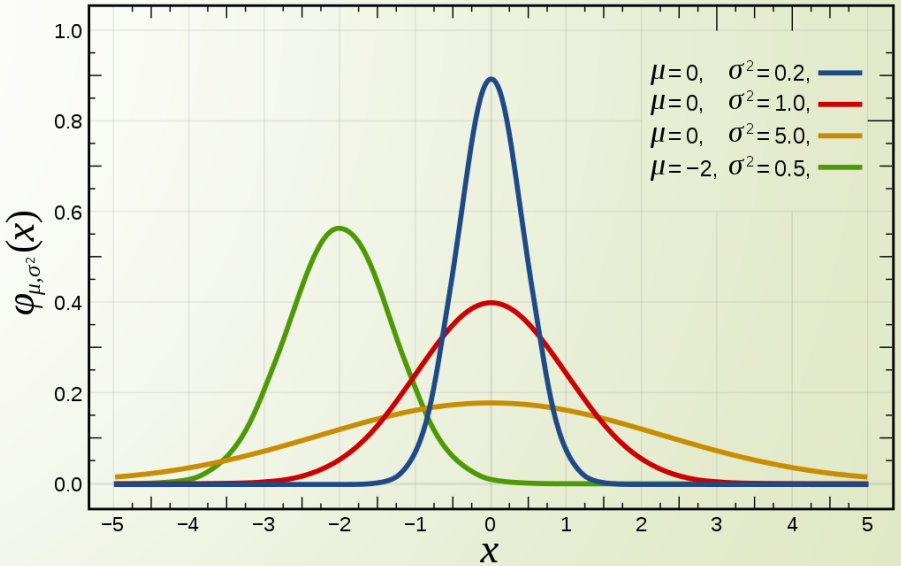
- **Removing noise**

Gaussian noise it is a statistical noise which is described by a probability density function. Random Gaussian function is added to the image function to generate this noise.



$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$





The previous types of noise are considered additive as they are added to the image $M = I + N$, other types of noise can be multiplicative.

Improving Image quality

- **Removing noise**

To deal with noise there are several types of filters. Let us first define the pixel neighborhood: is the pixel surrounding. Mainly there are two reasons behind the presence of noise in the image.

25	0	7	12	13	15	0	0
25	18	9	2	1	11	0	1
44	4	4	14	4	6	7	4
12	8	9	12	10	3	9	7
0	2	2	2	10	8	7	88
0	48	9	8	77	7	5	87
9	30	78	9	7	85	14	66
58	50	14	15	12	98	11	15

The pixels in yellow represent 3*3 neighborhood and pixels in blue represent 5*5 neighborhood of the pixel (4,5)

Improving Image quality

- **Removing noise using mean filter**

One way to remove noise is by summing up the values of the Pixel neighborhood, divide them on the neighborhood size and put the result in the concerned pixel

2	1	0
5	20	3
2	6	4

2	1	0
5	4,77	3
2	6	4

$$\frac{(2+1+0+5+20+3+2+6+4)}{9} = 4,77$$

Improving Image quality

- **Removing noise using mean filter**

Algorithm: Noise_Removal;

Input: image **I (N,N)**; Neighborhood $(2M+1) \times (2M+1)$
Output: image **R(N,N)**;
Begin

For i=M+1 to N-M
  For j=M+1 to N-M

$$R(i,j) = \frac{1}{(2M+1)^2} \sum_{k=-M}^{M} \sum_{l=-M}^{M} I(i+k, j+l)$$

  end
End

End.

Improving Image quality

- **Removing noise using mean filter**

Original Image

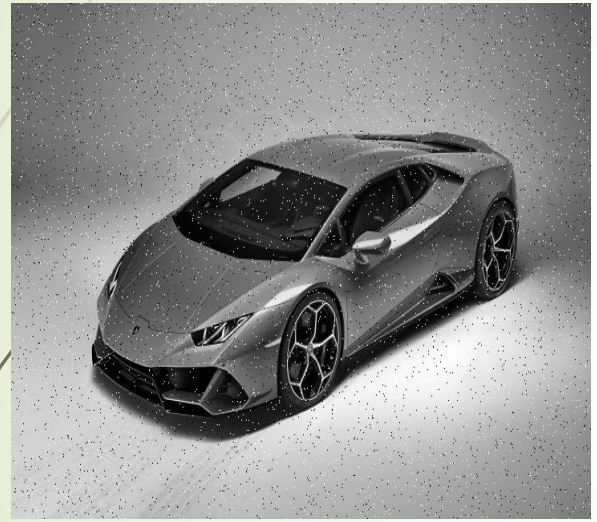
Image smoothed with
An average filter of
5x5

Image smoothed with
An average filter of
11x11

We can observe that the blur increases as the neighborhood size increases. Much smoothing lead to loss image details.

Improving Image quality

- **Removing noise using mean filter**



Original Image

Image smoothed with
An average filter of
5x5

Image smoothed with
An average filter of
11x11

We can observe that the blur increases as the neighborhood size increases. Much smoothing lead to loss image details.

Improving Image quality

- **Removing noise using mean filter**

The process of removing noise by browsing image top-down and left-right can be regarded as applying a filter on each image region. If we consider a neighborhood Of 3x3, the filter will looks like that

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

2	1	0
5	20	3
2	6	4

2	1	0
5	4,77	3
2	6	4

F * I = R

Applying a filter (called also kernel) on image is known as ***convolution***

Improving Image quality

- **Removing noise using mean filter**

Convolution is among the most basic operations in image processing

F=

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	2	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	2	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	2	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

Improving Image quality

- **Removing noise using mean filter**

F=

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	2	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	2	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	2	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

Improving Image quality

- **Removing noise using mean filter**

F=

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	2	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	2	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	2	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

Improving Image quality

- **Removing noise using mean filter**

Pixels to consider for a filter 3x3

2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	2	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

Pixels to consider for a filter 5x5

2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	2	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

Pixels to consider for a filter 7x7

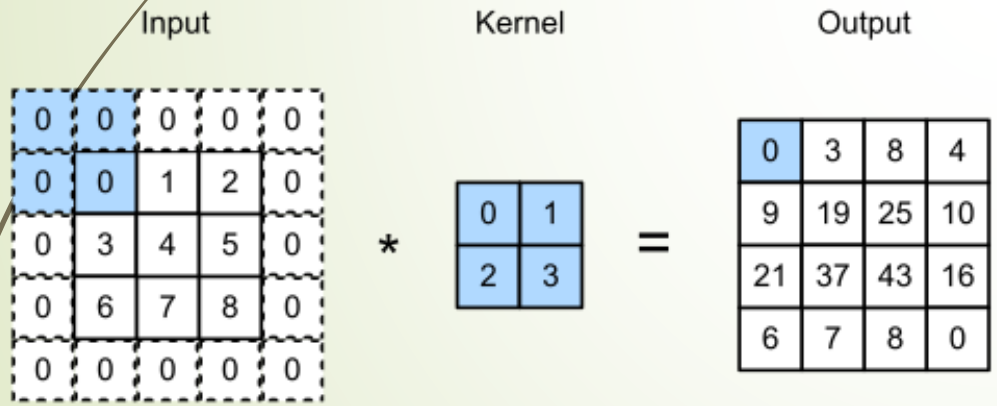
2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	2	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

Improving Image quality

- **Convolution versus cross-correlation**

Note that there is two different operations in this context namely ***convolution*** and ***cross-correlation***

Cross-correlation: measures the similarity between the patch and the image.



$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]\, F[i+u, j+v]$$

Most explanations of convolution are actually presenting cross-correlation.

Improving Image quality

- **Convolution versus cross-correlation**

Convolution: multiply two signals to produce a third signal. This process is similar to the cross-correlation, the only difference that the kernel is flipped 180°.

Kernel

5	6	7
8	9	0
4	3	1

Kernel (90°)

7	0	1
6	9	3
5	8	4

Kernel (180°)

1	3	4
0	9	8
7	6	5

-1,-1	0,1	1,-1
-1,0	0,0	1,0
-1,1	0,1	1,1

1,1	0,1	-1,1
1,0	0,0	-1,0
1,-1	0,1	-1,-1

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]\, F[i-u, j-v]$$

The convolution has a nicer mathematical properties.

Improving Image quality

- **Removing noise using Gaussian smoothing**

In the Gaussian smoothing, near pixels to the concerned pixel contribute more than far ones when computing the new pixel value. To do so, we use the Gaussian function which is parameterized by the mean and standard deviation $\boldsymbol{\sigma}$. The 1D / 2D Gaussian is given by

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} \times e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

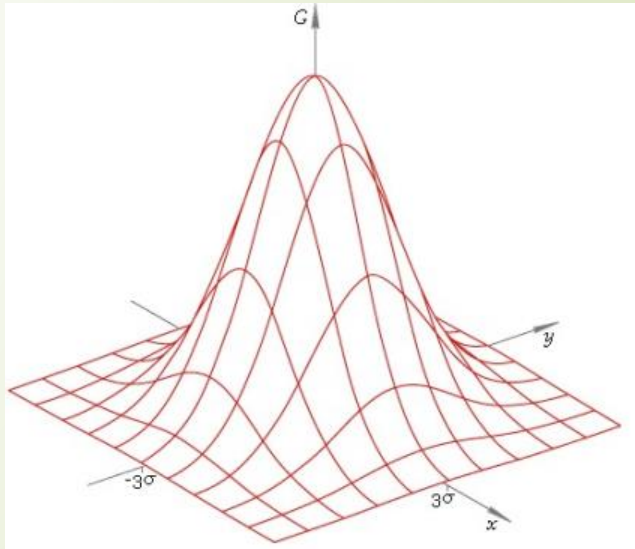
$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} \times e^{-\left(\frac{(x-\mu_x)^2}{2\sigma_x^2} + \frac{(y-\mu_y)^2}{2\sigma_y^2}\right)}$$

The centered and symmetric Gaussian is given by

$$g(x,y) = \frac{1}{2\pi\sigma^2} \times e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}$$

$$\sigma_x = \sigma_y = \sigma$$

$$\mu_x = \mu_y = 0$$



Improving Image quality

- **Removing noise using Gaussian smoothing**

As example, the following 5x5 filter is generated using a Gaussian function with $\boldsymbol{\sigma} = 0{,}5$ erate filters as follows

6.9625e-08	2.8089e-05	2.0755e-04	2.8089e-05	6.9625e-08
2.8089e-05	0.0113	0.0837	0.0113	2.8089e-05
2.0755e-04	0.0837	0.6187	0.0837	2.0755e-04
2.8089e-05	0.0113	0.0837	0.0113	2.8089e-05
6.9625e-08	2.8089e-05	2.0755e-04	2.8089e-05	6.9625e-08

We can see that this filter have the desired property, center pixel=0,6187, going far from it, the filter elements become less. Noting that the filter is normalized by dividing each elemnt on the filter sum.

Improving Image quality

- **Removing noise using Gaussian smoothing**

Algorithm: Noise_Removal;

Input: image **I (N,N)**; int **Filter_size**; double σ;
Output: image **R(N,N)**;
Begin

Generate a filter $\boldsymbol{F}$ with $\boldsymbol{\sigma}$ as parameter
Convolve $\boldsymbol{I}$ with $\boldsymbol{I} * \boldsymbol{F} = \boldsymbol{R}$

End.

Improving Image quality

- **Removing noise using Gaussian smoothing**

Original Image

Gaussian smoothing
With a 5x5 filter (σ=0,9)

Gaussian smoothing
With a 5x5 filter (σ=2,9)

We can observe that the blur increases as σ increases.

Improving Image quality

- **Removing noise using Gaussian smoothing**

Original Image

Gaussian smoothing
With a 5x5 filter (σ=0,9)

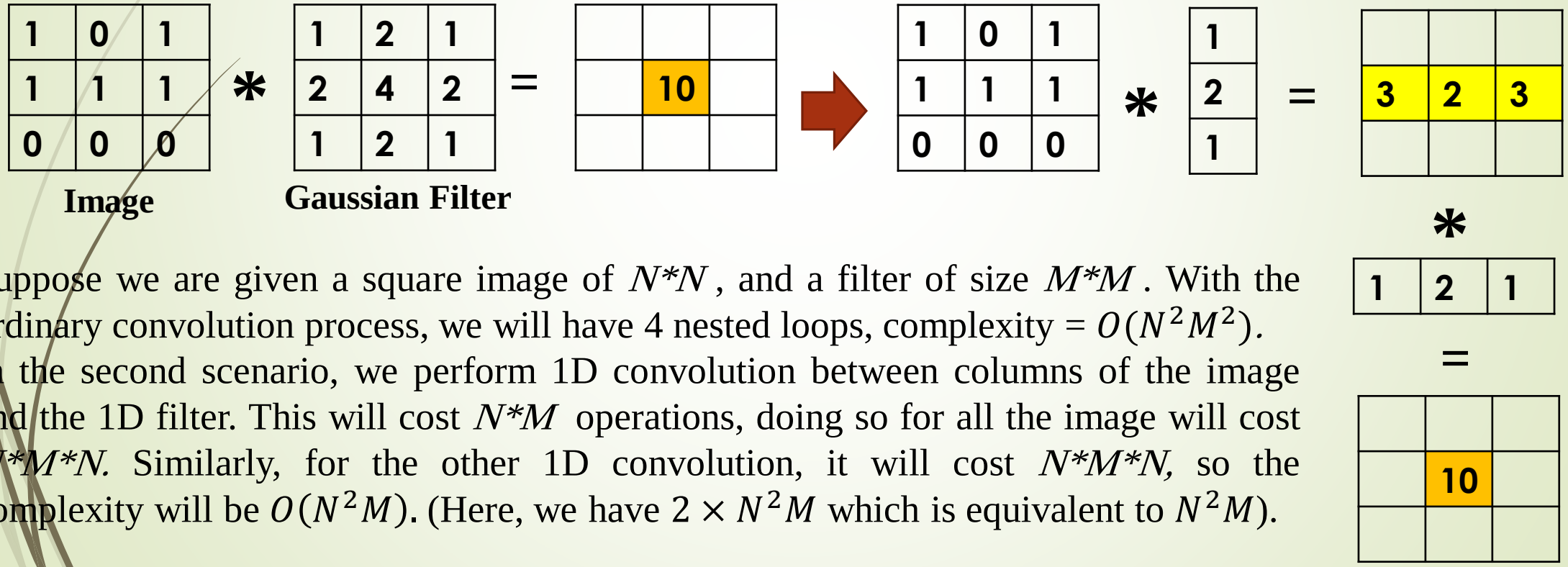
Gaussian smoothing
With a 5x5 filter (σ=2,9)

We can observe that the blur increases as σ increases.

Improving Image quality

- **Separability of filters**

To speed-up computation some filters can be separated.

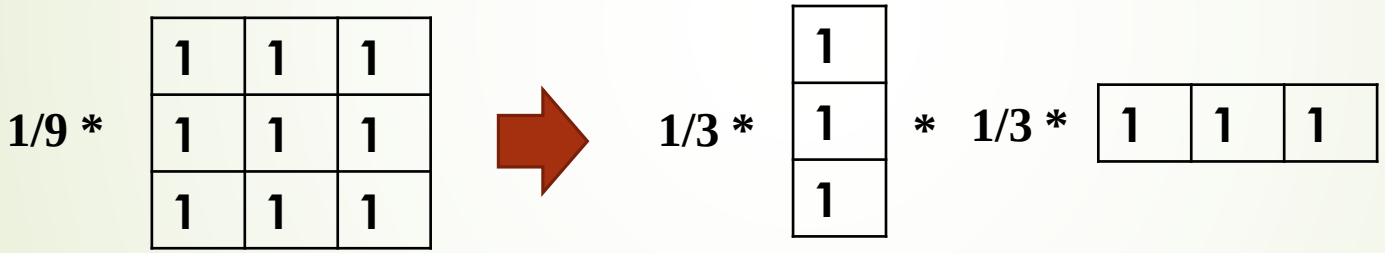


Suppose we are given a square image of $N*N$, and a filter of size $M*M$. With the ordinary convolution process, we will have 4 nested loops, complexity = $O(N^2M^2)$.
In the second scenario, we perform 1D convolution between columns of the image and the 1D filter. This will cost $N*M$ operations, doing so for all the image will cost $N*M*N$. Similarly, for the other 1D convolution, it will cost $N*M*N$, so the complexity will be $O(N^2M)$. (Here, we have $2 \times N^2M$ which is equivalent to N^2M).

Improving Image quality

- **Separability of filters**

The mean filter can also be separated

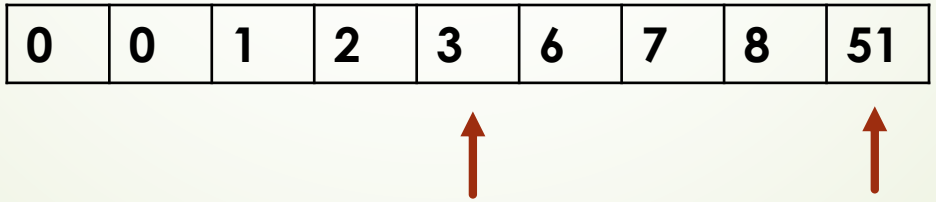


Improving Image quality

- **Removing noise using median filter**

Mean and Gaussian filters are linear filters as they can be expressed as $I \times F = N$. Where N is the resulting image F is the filter and I is the original image. However, linear filters are not well-suited for the salt-pepper noise. The median filter is a no-linear filter, which is well-suited to salt-pepper noise, and which preserves the image edges.

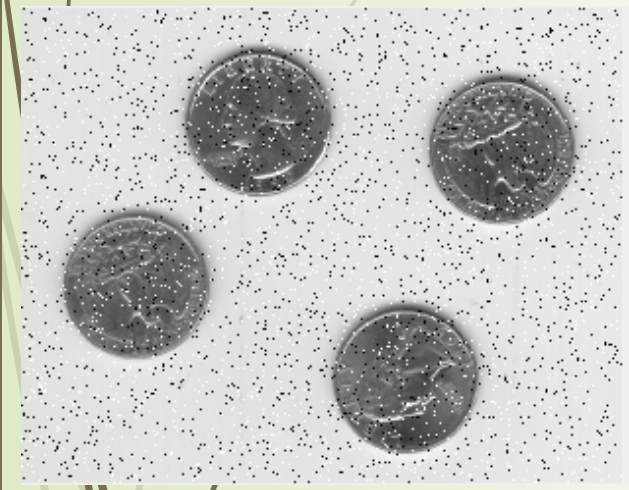
2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	3	6	1	5
0	3	1	51	2	1	4
8	1	0	0	8	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9



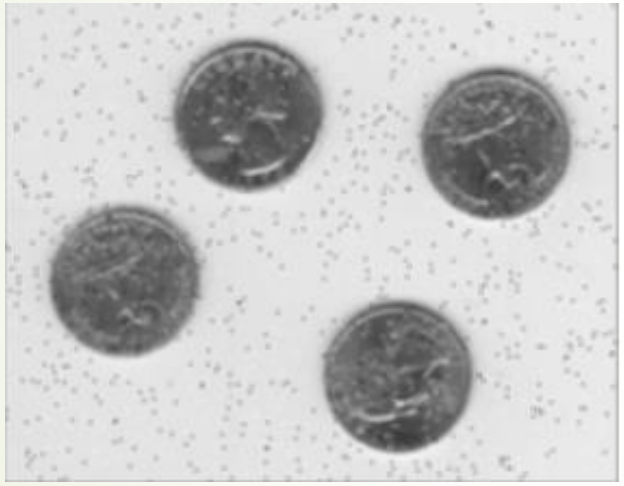
2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	3	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

Improving Image quality

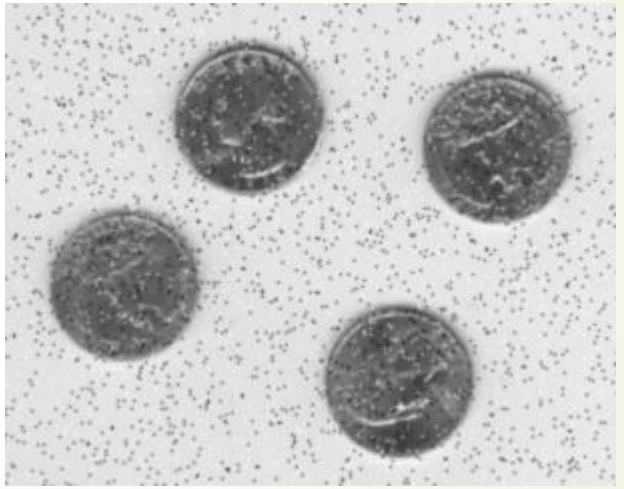
- **Removing noise using median filter**



Noisy image



Mean filter



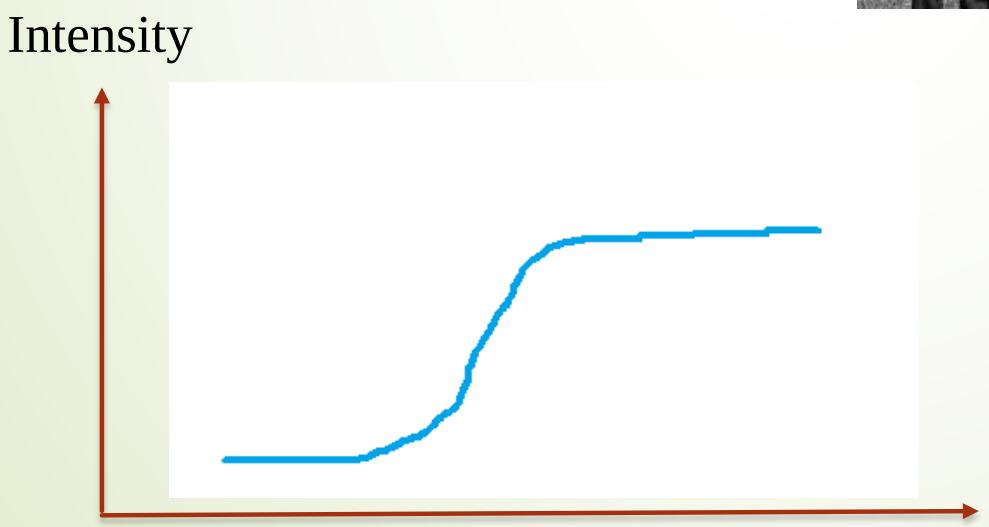
Gaussian filter

Median filter

Edge detection

- **What is edge detection**

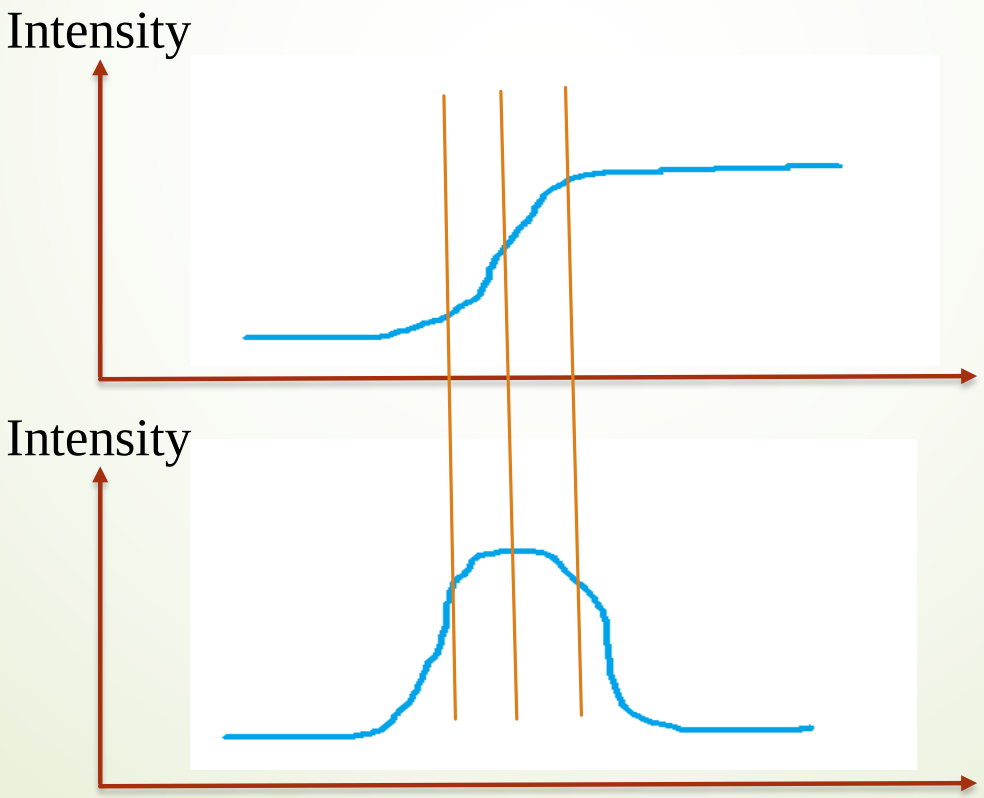
If we consider 1D i.e., one direction, either X or Y axis, edge refer to the change in intensity as shown by the following figure.



Edge detection

- **How to detect edge**

Edge can be located by detecting the inflexion point, which is the maximum in the first derivative.



Edge detection
Using image gradient

- **How to detect edge**

But how can we compute the first derivative of a specific image, knowing that image is considered as a 2D discrete function

Continuous case

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$$

Discrete case

$$I'(x) = \frac{I(x+1) - I(x-1)}{2}$$

Edge detection

- **How to detect edge**

But how can we compute the first derivative of a specific image, knowing that image is considered as a 2D discrete function

Continuous case

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$$

OR

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Discrete case

$$I'(x) = \frac{I(x+1) - I(x-1)}{2}$$

OR

$$I'(x) = I(x+1) - I(x)$$

This formula allows calculating the gradient at each pixel

Edge detection

- **Sobel operator**

Sobel filter is an approximation for computing the gradient (derivative)

$$G_X = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \qquad G_Y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

Detecting edge can be done by convolving the image with G_X and G_Y

Note that G_X detects **vertical** edge, while G_Y detects **horizontal** edge

Edge detection

- **Sobel operator**

Example of detecting vertical edge

50	50	200	200
50	50	200	200
50	50	200	200
50	50	200	200

$*$

-1	0	1
-2	0	2
-1	0	1

$=$

$$50 \times (-1) + 50 \times (-2) + 50 \times (-1) + 50 \times 0 + 50 \times 0 + 50 \times 0 + 200 \times (1) + 200 \times 2 + 200 \times (1) = 600$$

High values indicate the presence of an edge, while values close to zero indicate the inverse

Edge detection

- **Sobel operator**

If we take the same image and we consider the horizontal detector we get the following

50	50	200	200
50	50	200	200
50	50	200	200
50	50	200	200

$*$

-1	-2	-1
0	0	0
1	2	1

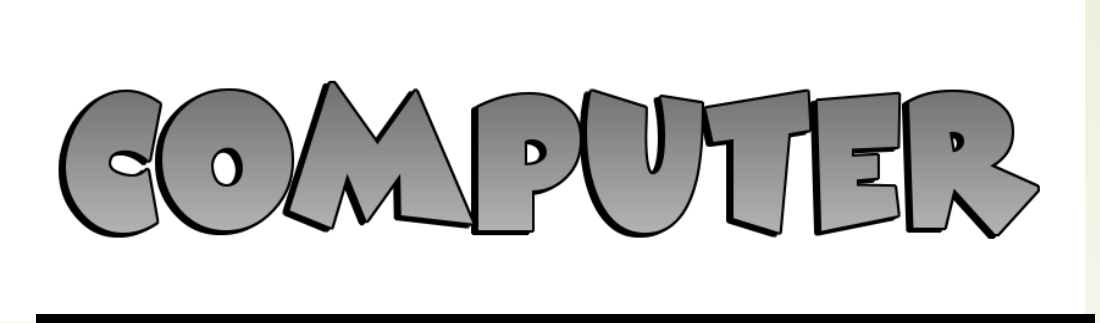
$=$

$$50 \times (-1) + 50 \times (0) + 50 \times (1) + 50 \times (-2) + 50 \times 0 + 50 \times 2 + 200 \times (-1) + 200 \times 0 + 200 \times (1) = 0$$

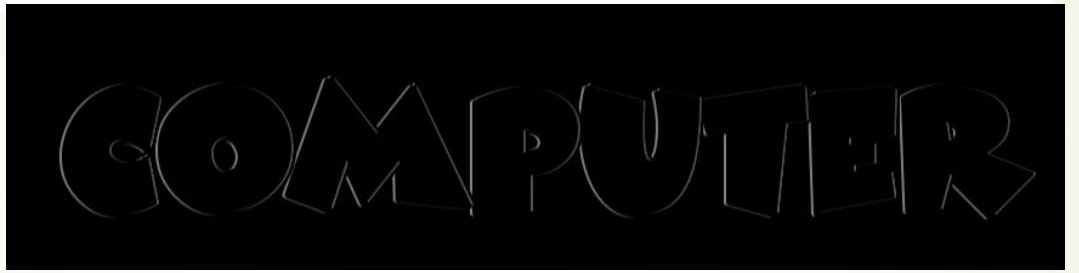
Here G_Y revealed that there is no horizontal edge in this image

Edge detection

Image

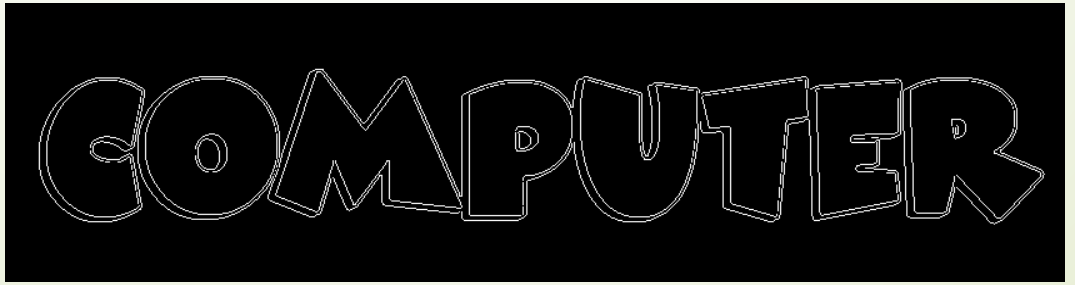


Vertical edge



Horizontal edge

Final edge



Edge detection

Algorithm: Edge_detection;

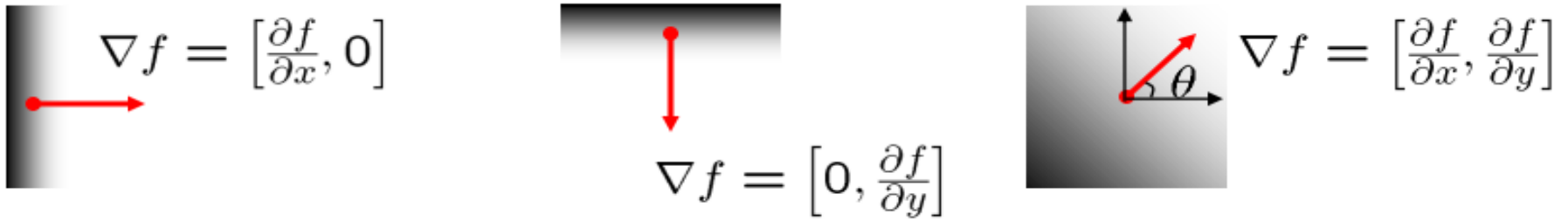
Input: image **I (N,N)**; Horizantal Filter **FH**; % example Sobel
Vertical Filter **FV**;
Output: image **R(N,N)**;
Begin

Compute the partial derevatives
- Convolve $\boldsymbol{I}$ with $\boldsymbol{I * FV = R_X}$
- Convolve $\boldsymbol{I}$ with $\boldsymbol{I * FH = R_Y}$

End.

Edge detection

- **Gradient magnitude and direction interpretation**

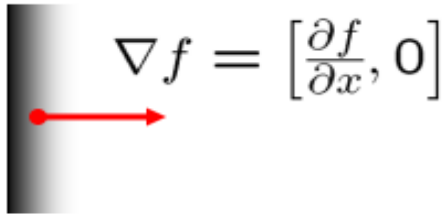


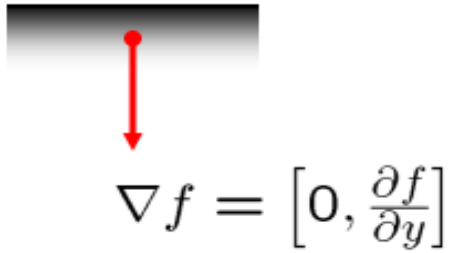
Compute gradient magnitude and orientation (for each pixel)

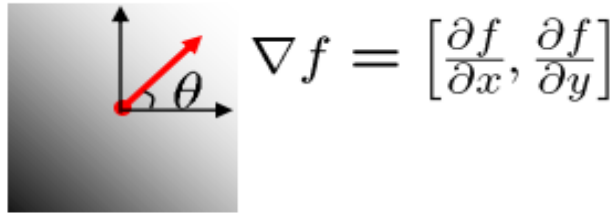
- Mag $(\boldsymbol{R}(\boldsymbol{x},\boldsymbol{y})) = \sqrt{\boldsymbol{R}_{\boldsymbol{X}}^{\boldsymbol{2}}(\boldsymbol{x},\boldsymbol{y}) + \boldsymbol{R}_{\boldsymbol{Y}}^{\boldsymbol{2}}(\boldsymbol{x},\boldsymbol{y})}$
- Orientation$\left(\boldsymbol{R}(\boldsymbol{x},\boldsymbol{y})\right) = \text{atan}\left(\frac{R_X(x,y)}{R_Y(x,y)}\right)$

Edge detection

- **Gradient magnitude and direction interpretation**







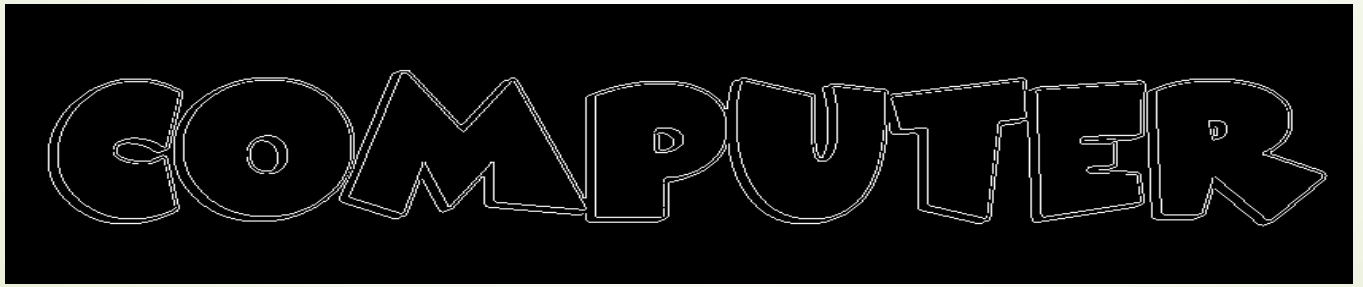
The gradient measure the change of image function. For instance, leftmost image change is only in X-axis, whereas, in the second image, change is in Y-axis. In the rightmost image, the gradient points in the direction of most rapid increase in intensity.

In brief, the magnitude of the gradient tells us how quickly the image is changing, while the direction of the gradient tells us the direction in which the image is changing.

Edge detection
Using image gradient

The following images can be considered as edge images

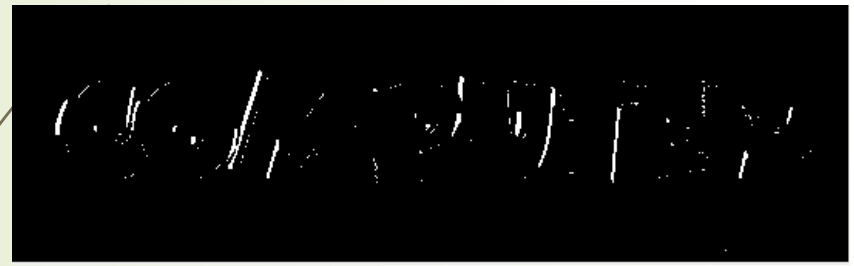
- Magnitude image (without performing any further processing), especially for images with a smooth background (e.g., the previous image, the world ‘computer’ is on a white background)



Edge detection
Using image gradient

The following images can be considered as edge images

- Taking pixels having gradient magnitude greater than a certain threshold is also an edge image



Keep Magnitude > 150

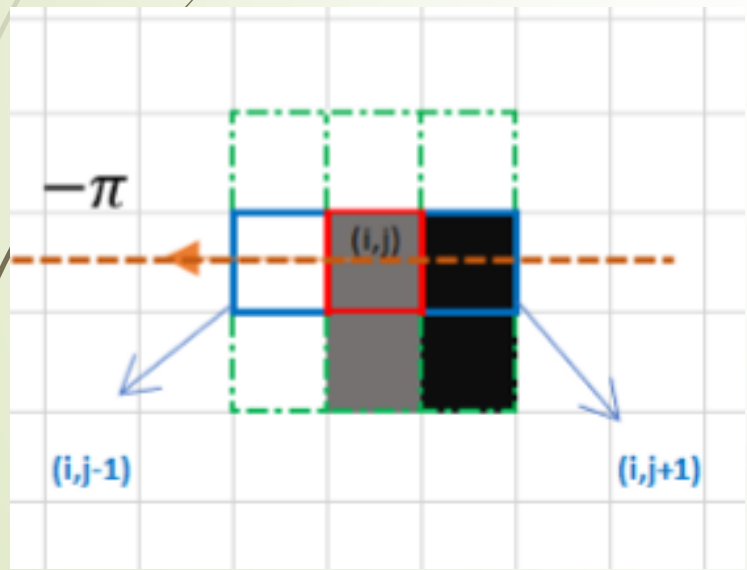
Keep Magnitude > 100

Keep Magnitude > 50

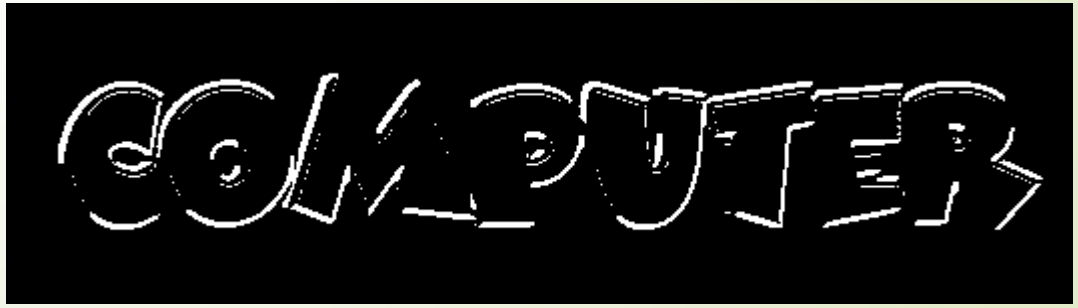
Keep Magnitude > 30

Edge detection
Using image gradient

- We can also take pixels having the maximum of gradient in the gradient orientation (direction).

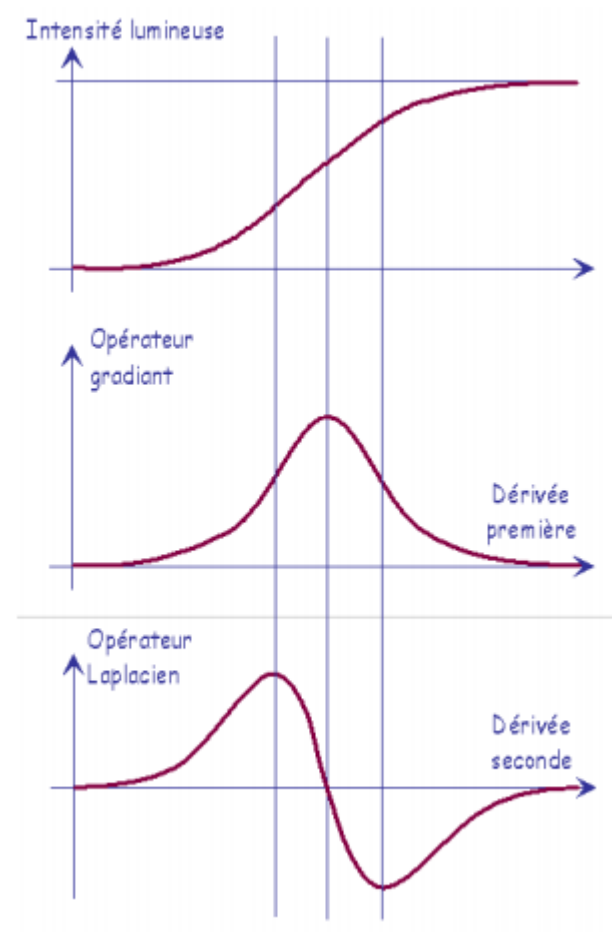


In this case, the direction is 180°, thus, we consider horizontal neighbors of the central pixel Img(i,j) i.e., left and right neighbors. We can notice that among the three pixels Img(i, j-1) is the most intense, and not the central pixel Img(i,j). Thus, we keep the most intense one, and we set the rest to 0.
Note that angle binning is required (to be seen later)



Edge detection
Using Laplacian of Gaussian

We can detect edge using the second derivatives of an image, then search for zero-crossings



Mathematically, second derivative is given by $\boldsymbol{I'(x) = I(x+1) - I(x)}$

$$f''(x) = \lim_{h->0} \frac{f'(x+h) - f'(x)}{h} \approx f'(x+1) - f'(x) =$$

$$f(x+2) - 2f(x+1) + f(x) \quad (h=1)$$

This approximation is centered to (X+1) (note that h converges to 0, and we have considered *h=1*, so we subtract the 1 we added), now, if we replace X by (X-1), we get

$$f''(x) \approx f(x+1) - 2f(x) + f(x-1)$$

Edge detection
Using Laplacian of Gaussian

By extending this formula to the case of 2D, we can see how we can extract a convolution filter

$$\Delta^2 f = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y}$$

$$\frac{\partial^2 f}{\partial^2 x} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

$$\frac{\partial^2 f}{\partial^2 y} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

$$\Delta^2 f = f(x+1,y) + f(x-1,y) - 2f(x,y) + f(x,y+1) + f(x,y-1) - 2f(x,y)$$

$$\Delta^2 \boldsymbol{f} = [\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{1},\boldsymbol{y}) + \boldsymbol{f}(\boldsymbol{x}-\boldsymbol{1},\boldsymbol{y}) + \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}+\boldsymbol{1}) + \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}-\boldsymbol{1})] - \boldsymbol{4f}(\boldsymbol{x},\boldsymbol{y})$$

$f(x-1,y-1)$	$f(x,y-1)]$	$f(x+1,y-1)$
$f(x-1,y)$	$f(x,y)$	$f(x+1,y)$
$f(x-1,y+1)$	$f(x,y+1)$	$f(x+1,y+1)$

0	**1**	**0**
1	**- 4**	**1**
0	**1**	**0**

Edge detection
Using Laplacian of Gaussian

However, this filter is sensitive to noise!

0	1	0
1	- 4	1
0	1	0

One way is to use Gaussian smoothing to alleviate noise before Computing the image derivatives

$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

What do you think if we can simultaneously preform the both Operations (i.e., derivation and smoothing), we can do so by Using **Laplacian of Gaussian (LoG)**

Edge detection
Using Laplacian of Gaussian

Laplacian of Gaussian (LoG) is given by

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\Delta G = \frac{1}{\sigma^2}\left(\frac{x^2+y^2}{\sigma^2} - 2\right)G(x,y)$$

In such a case, we generate a kernel using this function and we use it for Convolution. Here, Sigma = 1,4.

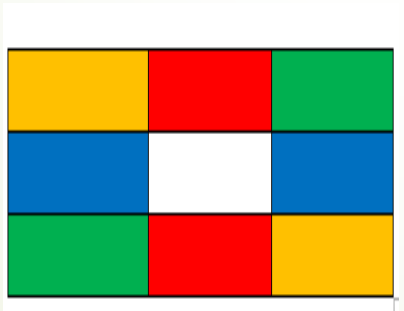
0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

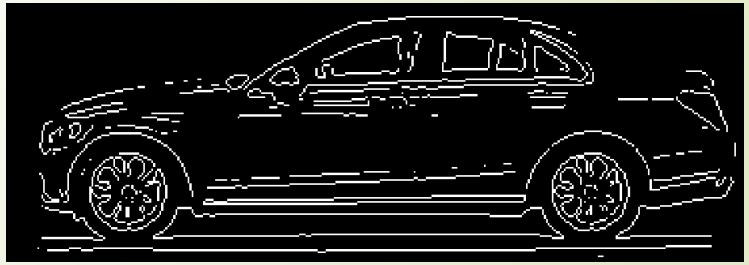
Edge detection
Using Laplacian of Gaussian

By convolving the input image using the Laplcian filter, we get another image. But how to detect edges! Edges = zero-crossings

zero-crossings: Using a 3x3 neighborhood centered at ***p***. A zero crossing at ***p*** implies that the signs of at least two of its opposing neighboring pixels must differ

- ❑ left/right
- ❑ Top/down or
- ❑ Diagonals





Edge detection
Using Laplacian of Gaussian

Algorithm: Edge_detection using LoG;

Input: image **I (N,N)**; **T:** threshold;
Output: image **R(N,N)**;
Begin

- $\boldsymbol{LoG_{XY}}$ = Generate the LoG filter using LoG function;
- Compute the second image derevative and smooth it
  - Convolve $\boldsymbol{I}$ with $\boldsymbol{I * LoG_{XY} = R}$
- Detect the zero-crossings in $\boldsymbol{R}$ (for each pixel)
- Keep only pixels for which $(\boldsymbol{R(x,y) > T})$

End.

Edge detection
Using Canny algorithm

Canny algorithm is among the commonly used algorithms for edge detection

- The first step in Canny is to reduce noise using Gaussian smoothing

Here, Gaussian Smoothing with Sigma = 0,9

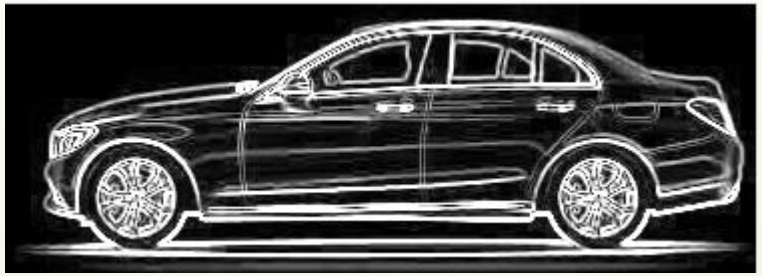
Edge detection
Using Canny algorithm

- The second step is to calculate the image gradient magnitude/direction. Magnitude can be calculated using Sobel operator

$$\text{Mag}\,(\boldsymbol{R}(\boldsymbol{x},\boldsymbol{y})) = \sqrt{\boldsymbol{R}_{\boldsymbol{X}}^{\boldsymbol{2}}(\boldsymbol{x},\boldsymbol{y}) + \boldsymbol{R}_{\boldsymbol{Y}}^{\boldsymbol{2}}(\boldsymbol{x},\boldsymbol{y})}$$

$$\text{Orientation}\big(\boldsymbol{R}(\boldsymbol{x},\boldsymbol{y})\big) = \text{atan}\left(\frac{R_X(x,y)}{R_Y(x,y)}\right)$$

Original Image



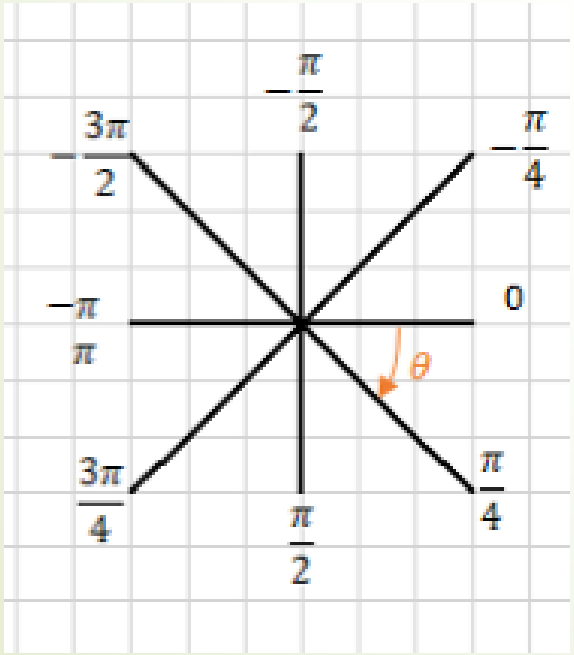
Gradient magnitude

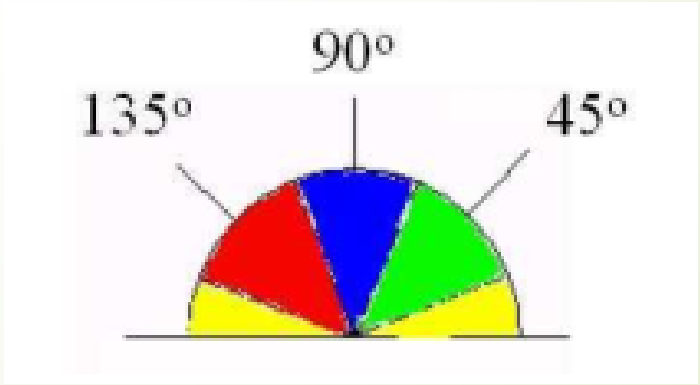
Gradient direction

Edge detection
Using Canny algorithm

- **Binning:** angles are rounded to 0°, 45°, 90° and 135°

Thus, $\boldsymbol{\theta} = 180°$ will be $\boldsymbol{\theta'} = 0°$ (bin 1), $\boldsymbol{\theta} = 225°$ will be $\boldsymbol{\theta'} = 45°$. If $\boldsymbol{\theta}$ is negative, thus add to it 180.



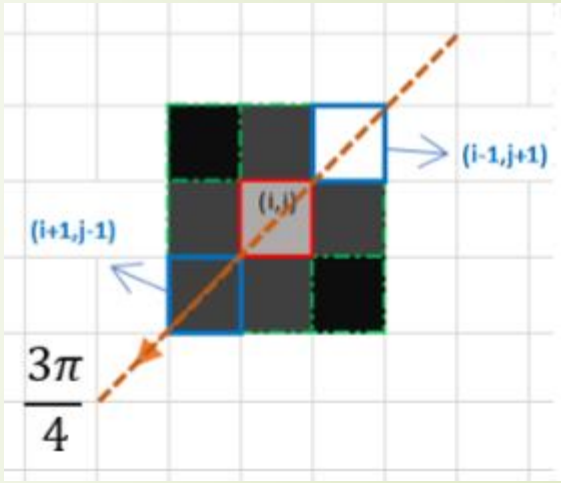


Edge detection
Using Canny algorithm

❑ **Non-Maximum Surpression**: keeps only those pixels on an edge with the highest gradient magnitude

So, three pixels in a 3×3 around pixel (x, y) are examined:

- If $\theta'(x, y) = 0°$, then the pixels $(x + 1, y)$, (x, y), and $(x - 1, y)$ are examined.
- If $\theta'(x, y) = 90°$, then the pixels $(x, y + 1)$, (x, y), and $(x, y - 1)$ are examined.
- If $\theta'(x, y) = 45°$, then the pixels $(x + 1, y + 1)$, (x, y), and $(x - 1, y - 1)$ are examined.
- If $\theta'(x, y) = 135°$, then the pixels $(x + 1, y - 1)$, (x, y), and $(x - 1, y + 1)$ are examined.



Here I(i-1, j-1) is kept

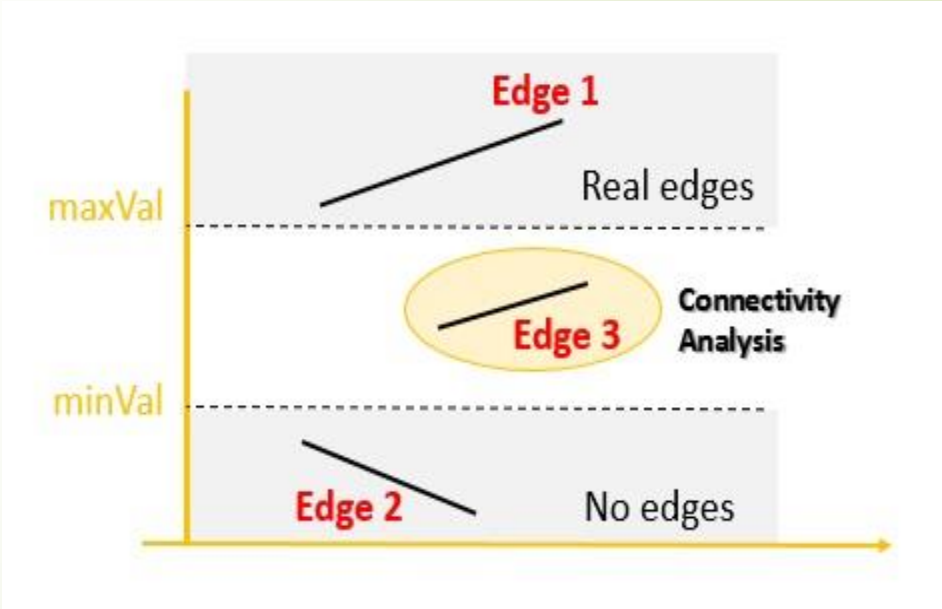
If pixel (x, y) has the highest gradient magnitude of the three pixels examined, it is kept as an edge. If one of the other two pixels has a higher gradient magnitude, then pixel (x, y) is not on the "center" of the edge and should not be classified as an edge pixel.

Edge detection
Using Canny algorithm

- **Hysteresis Thresholding**:

Some of the edges detected by Steps 1–3 will not actually be valid, but will just be noise. We would like to filter this noise out. Eliminating pixels whose gradient magnitude D falls below some threshold removes the worst of this problem, but it introduces a new problem.

A simple threshold may actually remove valid parts of a connected edge, leaving a disconnected final edge image. This happens in regions where the edge's gradient magnitude fluctuates between just above and just below the threshold. ***Hysteresis*** is one way of solving this problem. Instead of choosing a single threshold, two thresholds thigh and Tlow are used. Pixels with a gradient magnitude (D < Tlow) are discarded immediately. However, pixels with (Tlow ≤ D < Thigh) are only kept if they form a continuous edge line with pixels with high gradient magnitude (i.e., above Thigh).



Edge detection
Using Canny algorithm

- **Hysteresis Thresholding**

- If pixel (x, y) has gradient magnitude less than t_{low} discard the edge (write out black).
- If pixel (x, y) has gradient magnitude greater than t_{high} keep the edge (write out white).
- If pixel (x, y) has gradient magnitude between t_{low} and t_{high} and any of its neighbors in a 3×3 region around it have gradient magnitudes greater than t_{high}, keep the edge (write out white).
- If none of pixel (x, y)'s neighbors have high gradient magnitudes but at least one falls between t_{low} and t_{high}, search the 5×5 region to see if any of these pixels have a magnitude greater than t_{high}. If so, keep the edge (write out white).
- Else, discard the edge (write out black).

Original Image

Edge detection using Canny