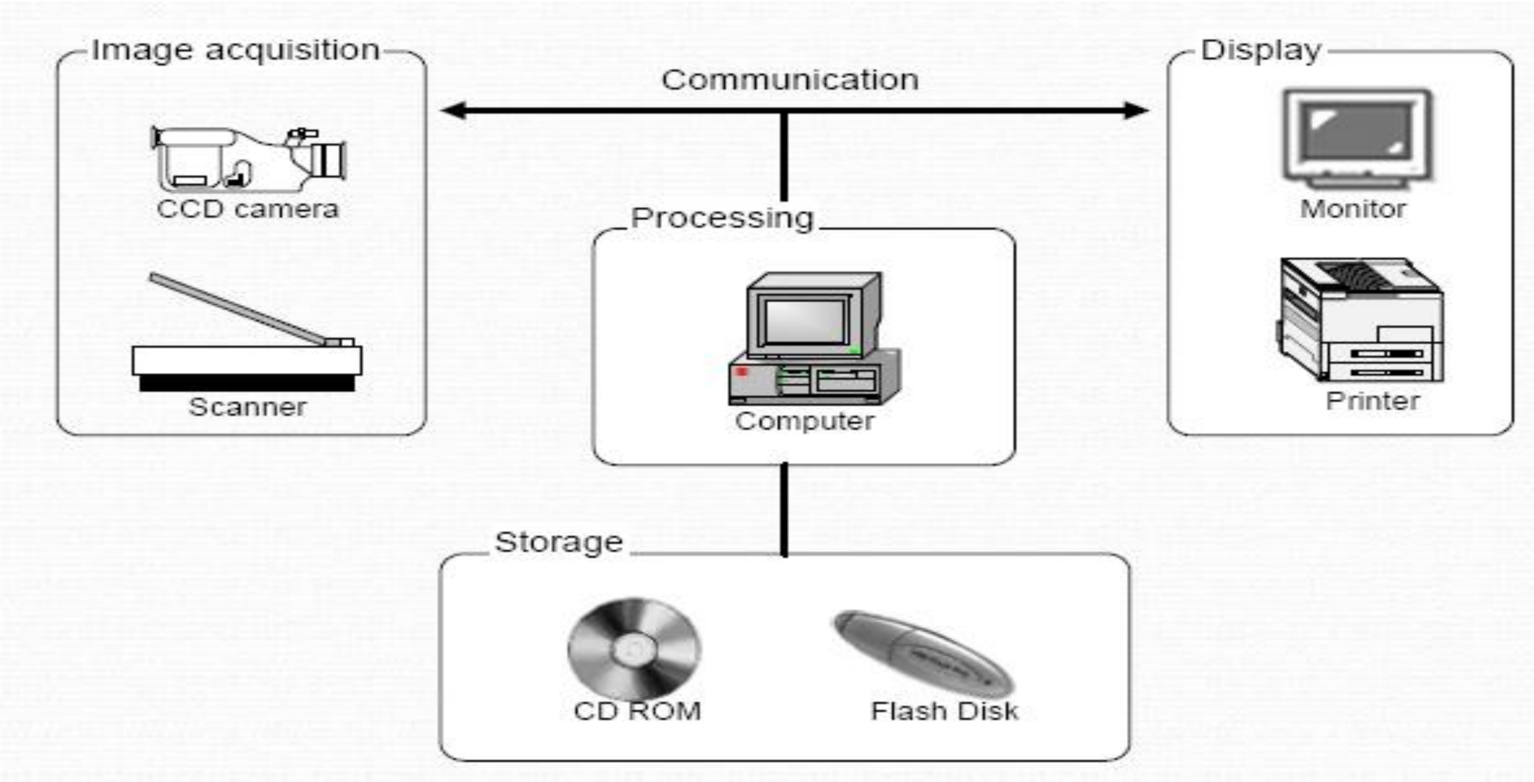
Computer vision

IMAGE PROCESSING

Tara Qadr
2024-2025
3rd Stage
Computer department
College of science
University of Sulaimani

Image Processing System



Digital Image Analysis System

- **A 2D image is nothing but a mapping from a region to a matrix**
- **A Digital Image Processing System consists of** (ASP-CD)

(یه اسپ دهان)

1. **Acquisition** – scanners, digital camera, ultrasound, X-ray, MRI, PMT
2. **Storage** – HD (120GB), CD (700MB), DVD (4.7GB), Flash memory (512MB~4GB), 3.5" floppy diskettes, i-pod, …
3. **Processing Unit** – PC, Workstation, PC-cluster
4. **Communication** – telephone lines, cable, wireless, …
5. **Display** – LCD monitor, laser printer, laser-jet printer

Types of Image Operations

The types of operations that can be applied to digital images to transform an input image a[m,n] into an output image b[m,n] (or another representation) can be classified into three categories

Operations	characterizations	Generic Complexity/Pixel
1 - Point	the output value at a specific coordinate is dependent only on the input value at that same coordinate	constants
2 - Local	the output value at a specific coordinate is dependent on the input values in the *neighborhood* of that same coordinate	P^2
3 - Global	the output value at a specific coordinate is dependent on all the values in the input image	N^2

Types of image operations. Image size = N x N; neighborhood size = P x P

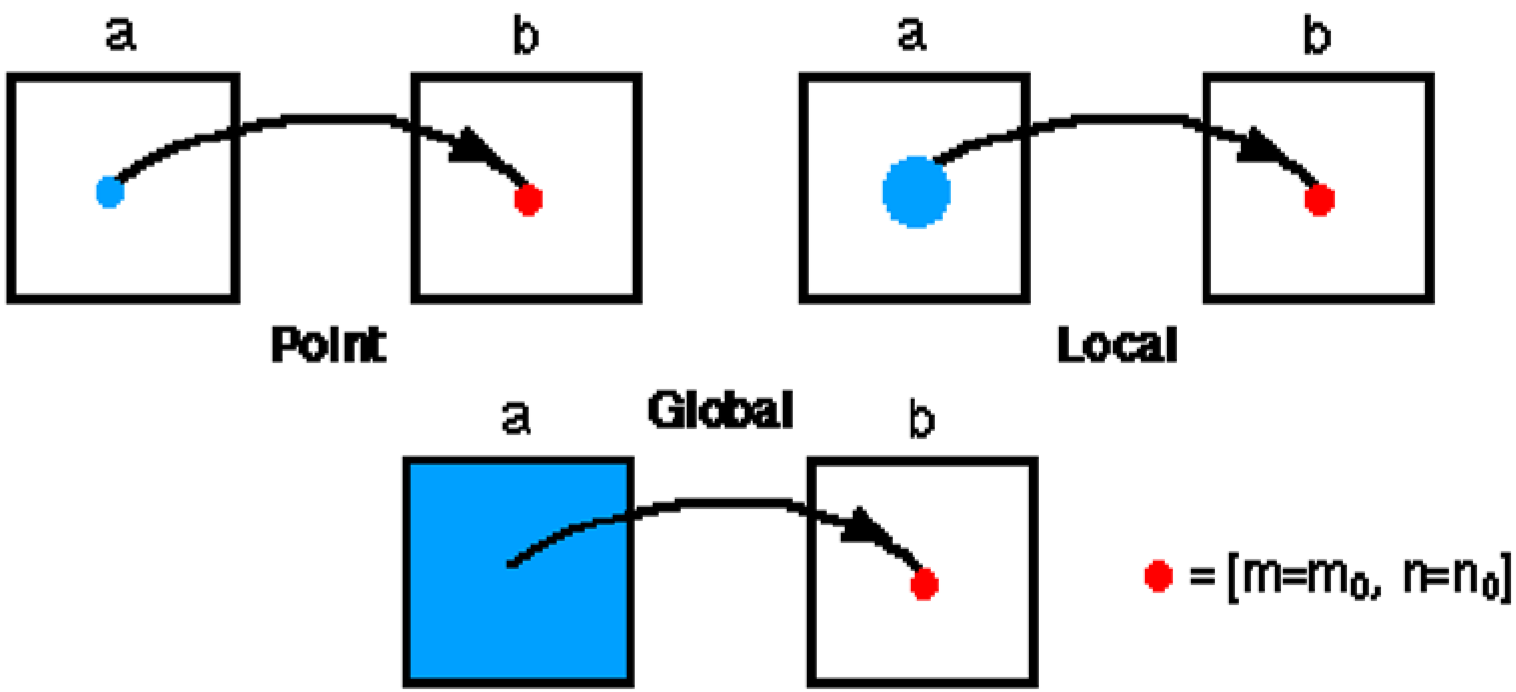
a Input

5	8	12	3
2	17	15	10
4	35	6	2
1	9	13	7

b Output

Types of Image Operations

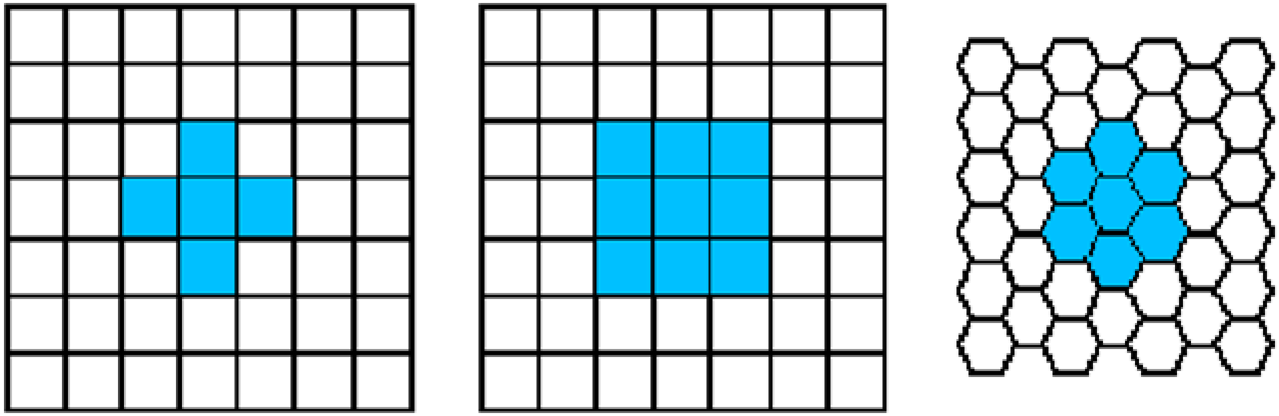
- An input image $a[m,n]$ and an output image $b[m,n]$ (or another representation)



Neighborhoods

- It is important to understand how images can be sampled and how that relates to the various neighborhoods that can be used to process an image.

- Some of the most common neighborhoods are the 4-connected neighborhood and the 8-connected neighborhood in the case of rectangular sampling and the 6-connected neighborhood in the case of hexagonal sampling



4-connected neighborhood 8-connected neighborhood 6-connected neighborhood

Classification of Image Operations

- **Spatial domain methods:** PAG-F (transformation)
    * Point Processing Transformations
    * Area/Mask Processing Transformations
    * Geometric Transformations
    * Frame Processing Transformations

- **Frequency domain methods**
    * low-pass Filter (LPF)
    * high-pass filter (HPF)

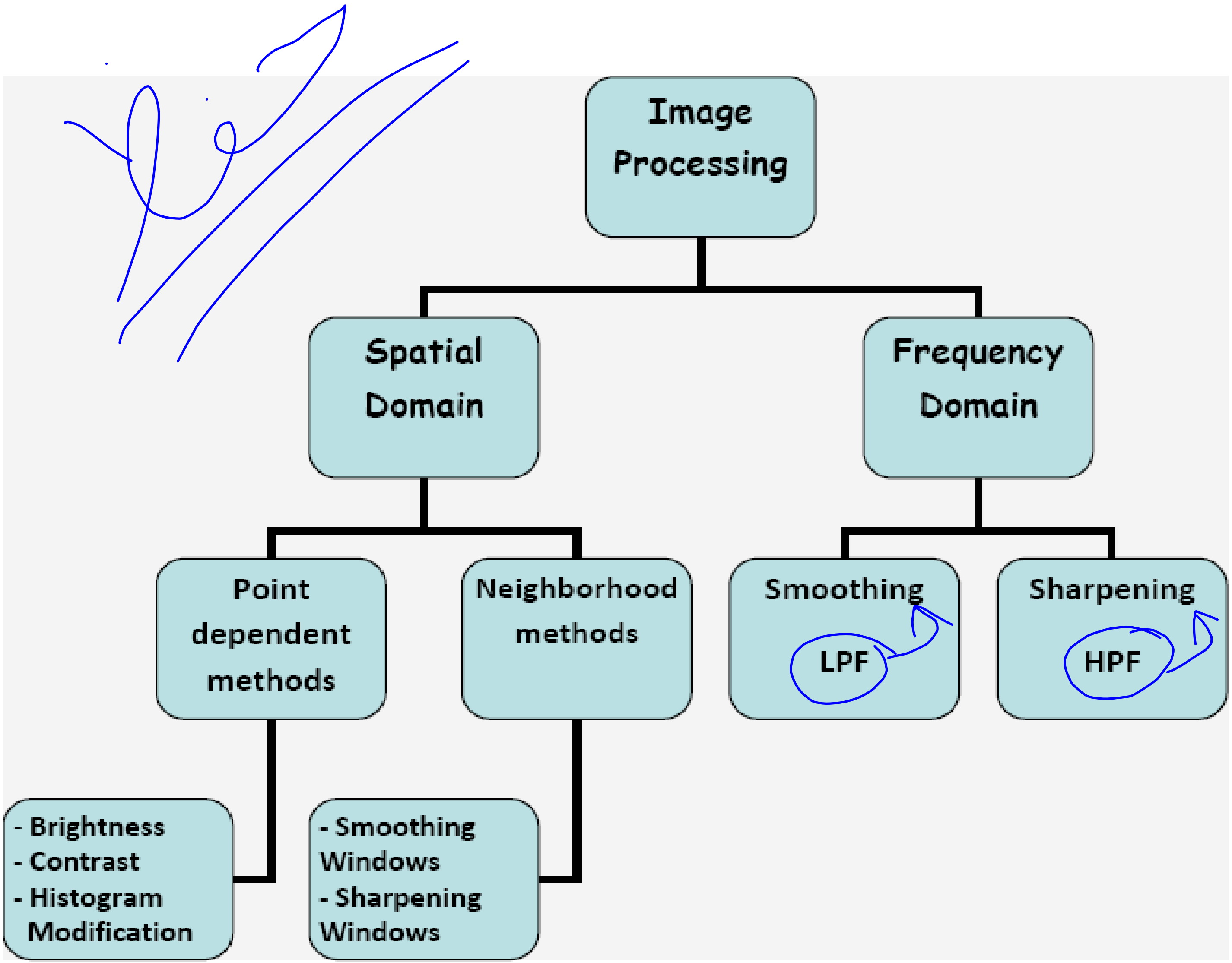
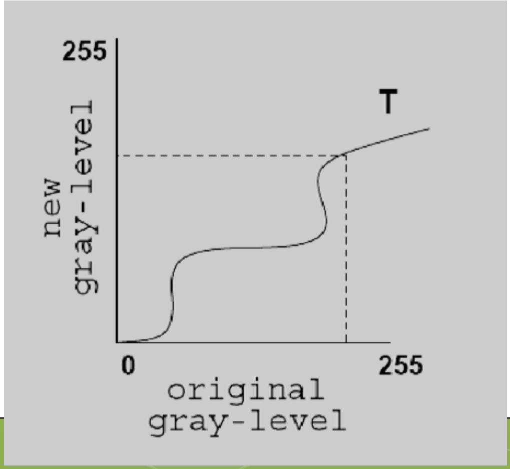


Image Processing
Spatial Domain
Frequency Domain
Point dependent methods
Neighborhood methods
Smoothing
LPF
Sharpening
HPF
- Brightness
- Contrast
- Histogram Modification
- Smoothing Windows
- Sharpening Windows

Point processing

The most primitive, yet essential, image processing operations.

- Intensity transformations that convert an old pixel into a new pixel based on some predefined function.
- They operate on a pixel based solely on that pixel's value.
- Used primarily for contrast enhancement.



Point processing (count..)

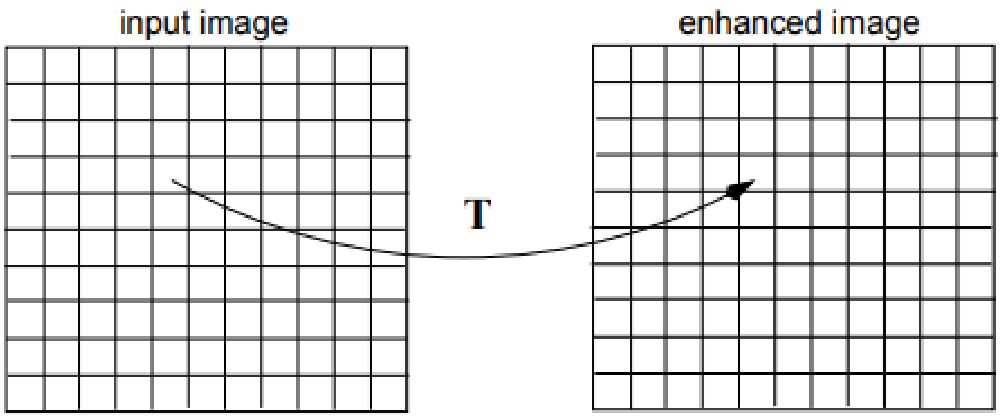
- Simple gray level transformations:
  - Image negatives
  - Log transformations
  - Power-law transformations
  - Contrast stretching
  - Gray-level slicing
  - Bit-plane slicing
- Histogram processing
  - Histogram equalization
  - Histogram matching (specification).
- Arithmetic/logic operations

point processing	Mask processing
The simplest kind of range transformations are these independent of position (x,y), For each original image intensity value I, function t() returns a transformed intensity value t(I). this is called as point processing.	The general approach in that is the value of g(x,y) is determined by the values of M is a predefined neighborhood of (x,y). Typical values of M range from 3 to 10. This technique is called an "Mask processing" or filtering
In point processing pixels can be treated independently of their neighbors.	In mask processing pixels are dependent on their neighbors.
Point processing are gamma correction, window center correction and histogram equalization.	Mask processing are mean, Gaussian, Median filters and image gradients.

GWh

MGMI

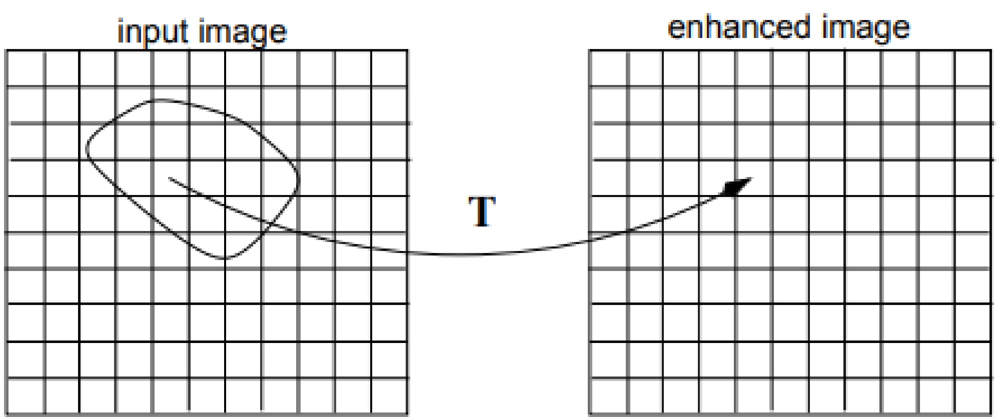
Point Processing Methods



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

T operates on 1 pixel

Area or Mask Processing Methods



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

T operates on a neighborhood of pixels

Point-Based Image Arithmetic

Image Operations: C [x, y] = f (A [x, y], B [x, y])

- Operates on each corresponding point from two (or more) images
- (Usually) requires that both images have the same dimensions.

Arithmetic operations are :

simple mathematical operations can be performed on the elements of two images

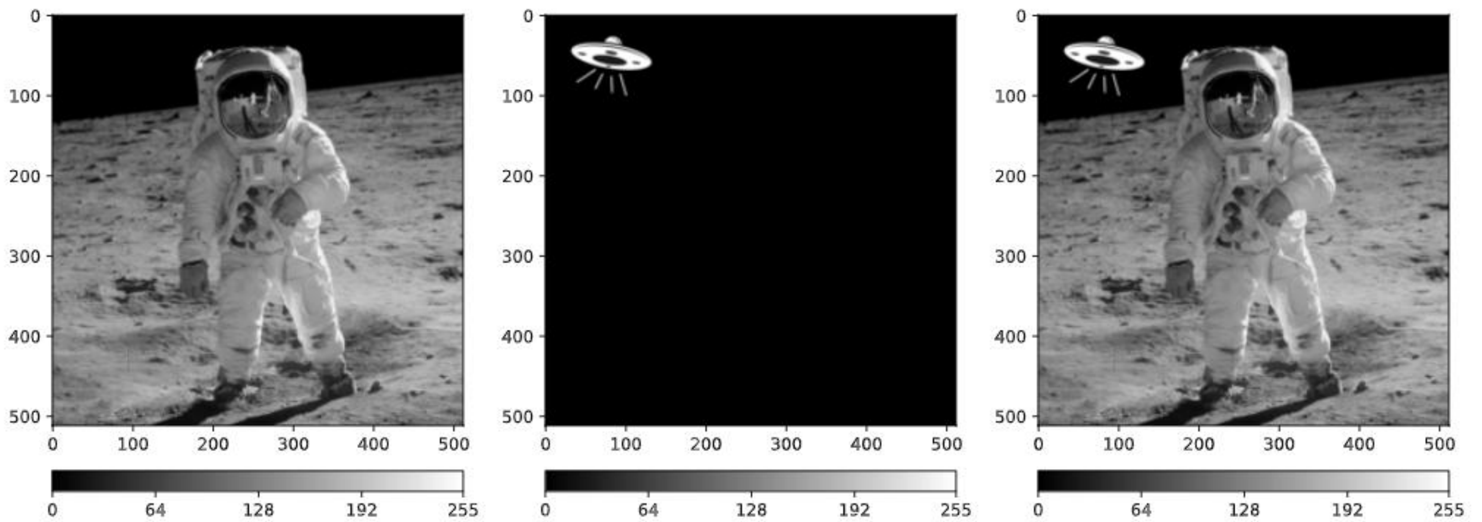
Addition ,Averaging, Subtraction and Multiplication , division

1 2 3 4

1-Image Addition

The addition of two images **f** and **g** of the same size results in a new image **h** of the same size whose pixels are to the sum of the pixels in the original images:

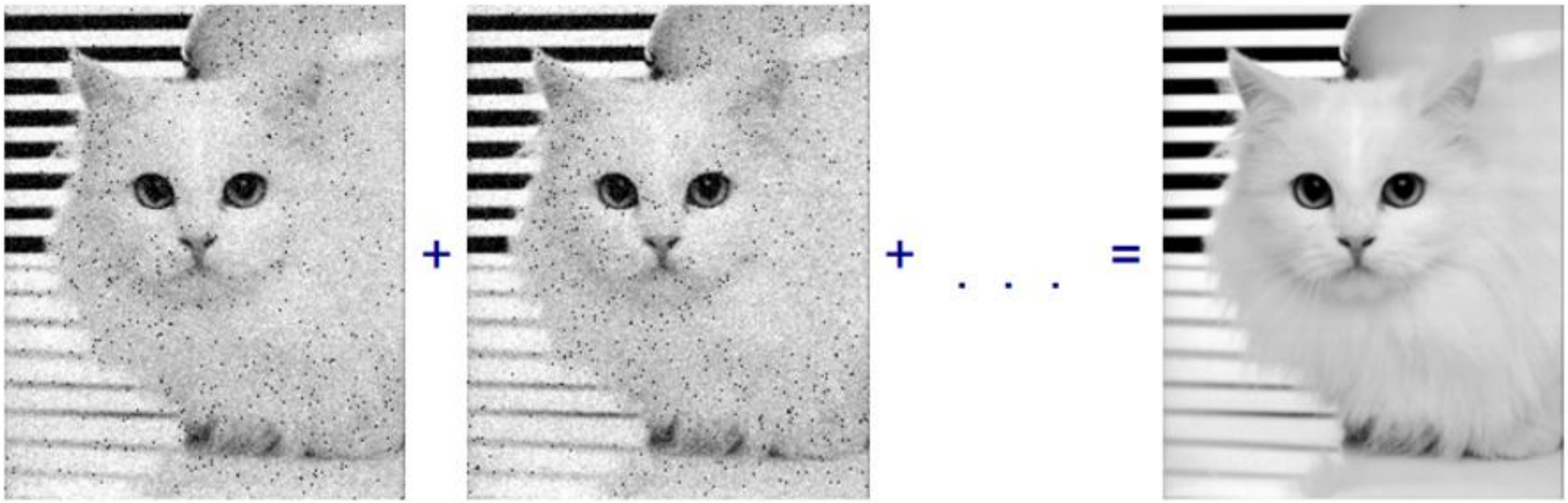
Used to create double-exposures



2-Image Averaging

Average multiple images (frames) of the same scene together

- Useful for removing noise



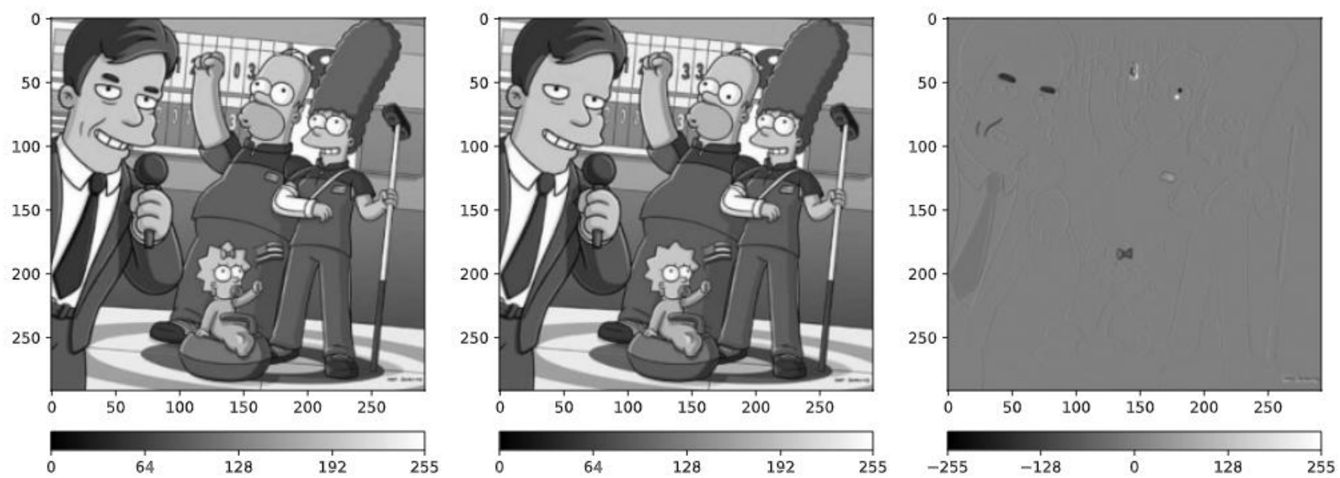
3-Image Subtraction

- The subtraction of two images is used for example to (1) detect changes (2)
- Use differencing to identify motion in an otherwise unchanging scene (object motion, not camera motion).
- Use in (3) Angiography Medical imaging technique

$$\forall m,n, \quad h(m,n) = f(m,n) - g(m,n)$$

or

$$\forall m,n, \quad h(m,n) = |f(m,n) - g(m,n)|$$

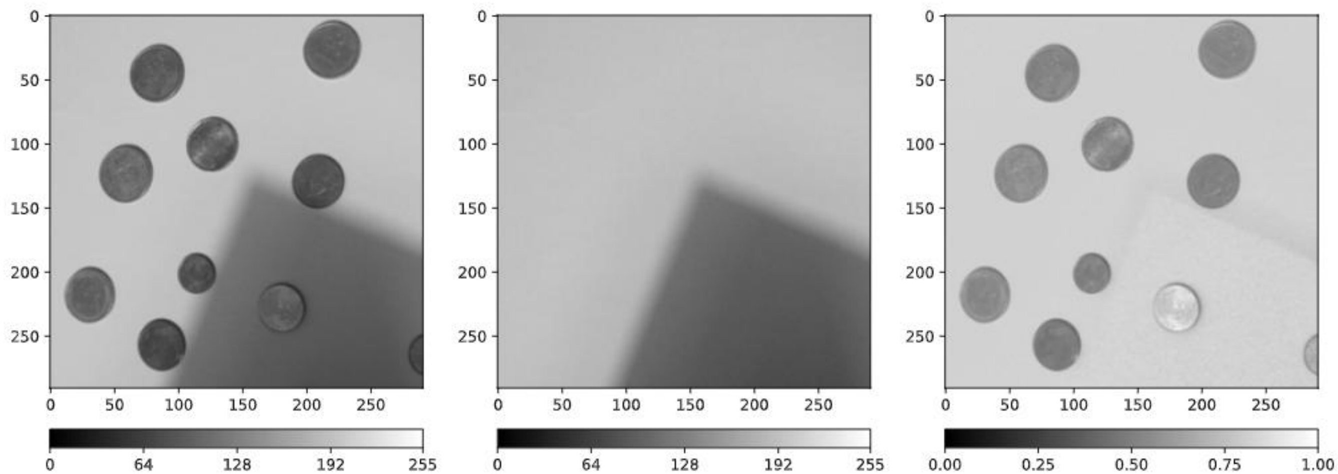


4-Division

الأضاءة الغير متجانسة

The division of two images is used to correct non-homogeneous illumination.

$$\forall m, n, \quad h(m,n) = \frac{f(m,n)}{g(m,n)}.$$



5-Multiplication

Useful for masking and alpha blending: $C[x, y] = A[x, y] \times B[x, y]$

ابراز او اخفاء   دمج

Single Image Point Operations

- Simplest kind of image enhancement
  - Also called **level operations**
  - Process each point independently of the others
- Remaps each sample value:

$g' = f(g)$ where

- g is the input value (gray level)
- g′ is the new (processed) result
- f is a level operation

Operations are :

- Adding a Constant
- Amplification (Gain)
- Linear Level Operators
- Negative
- Thresholding

g

5	8	12	3
2	17	15	10
4	35	6	2
1	9	13	7

input

g′

7	8	12	3
2	17	15	10
4	35	6	2
1	9	13	7

output

Single Image Point Operations (count..)

- **Adding a brightness**

Simplest level operation:

$$f(g) = g + b$$

for some constant (bias) b

Gray level values

0 ……70…………….. 255

- b > 0 Brighter Image
- b < 0 Darker Image

- **Amplification (Gain)**

Another simple level operation is amplification (multiplication):

$$f(g) = ag$$

for some constant gain (amplification) a

- a > 1 Amplifies signal (louder, more contrast)
- a < 1 Diminishes signal (softer, less contrast)

Single Image Point Operations (count..)

- **Linear Level Operators**

Linear operator combine gain (multiplication) and offset (addition):

f(g) = ag + b

where - **a** is the gain - **b** is the bias

5	8	12	3
2	255	15	10
4	35	0	2
1	9	13	7

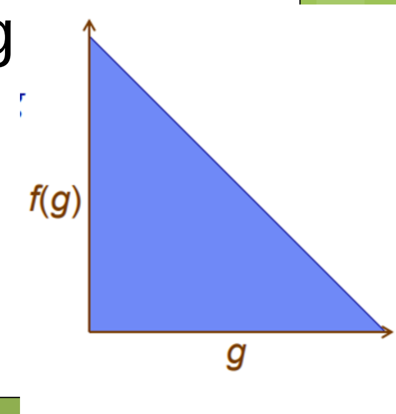
- **Negative**

Computing the "negative" of the signal/image:

f(g) = –g

• Or, to keep the range positive: f(g) = gmax – g

where g ∈ [0, gmax] ƒThis is simply a line with slope = –1



Single Image Point Operations (count..)

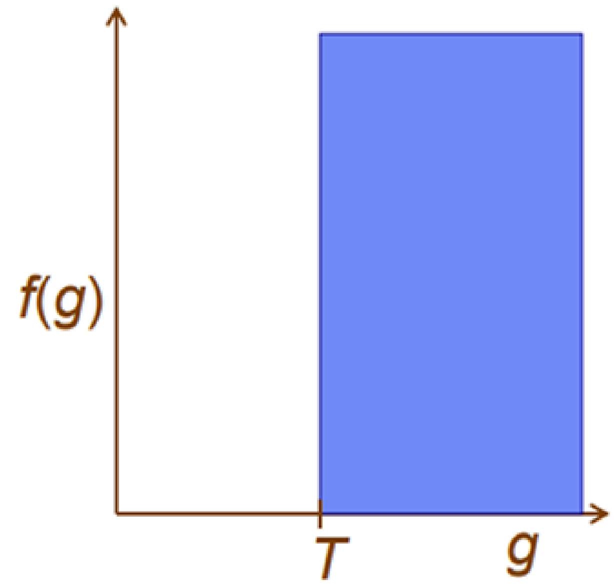
- **Thresholding**

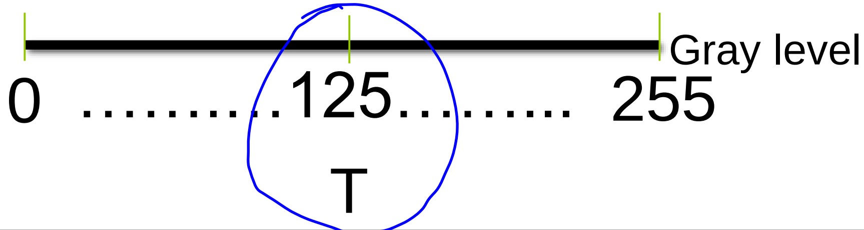
Thresholding a signal:

$$f(g) = \begin{cases} 0 & \text{if } g < T \\ 1 & \text{otherwise} \end{cases}$$

for some intensity threshold T

50	75	12	170
2	255	15	166
240	35	150	20
40	9	13	200





Single Image Point Operations (count..)

- **Logarithms**

Used to consider relative changes g1/g2 instead of absolute ones g1 – g2:

f(g) = log(g)

- Useful when the dynamic range is large

Examples: *f*

Apparent brightness

f Richter scale *f*

Human vision

- **Exponential**

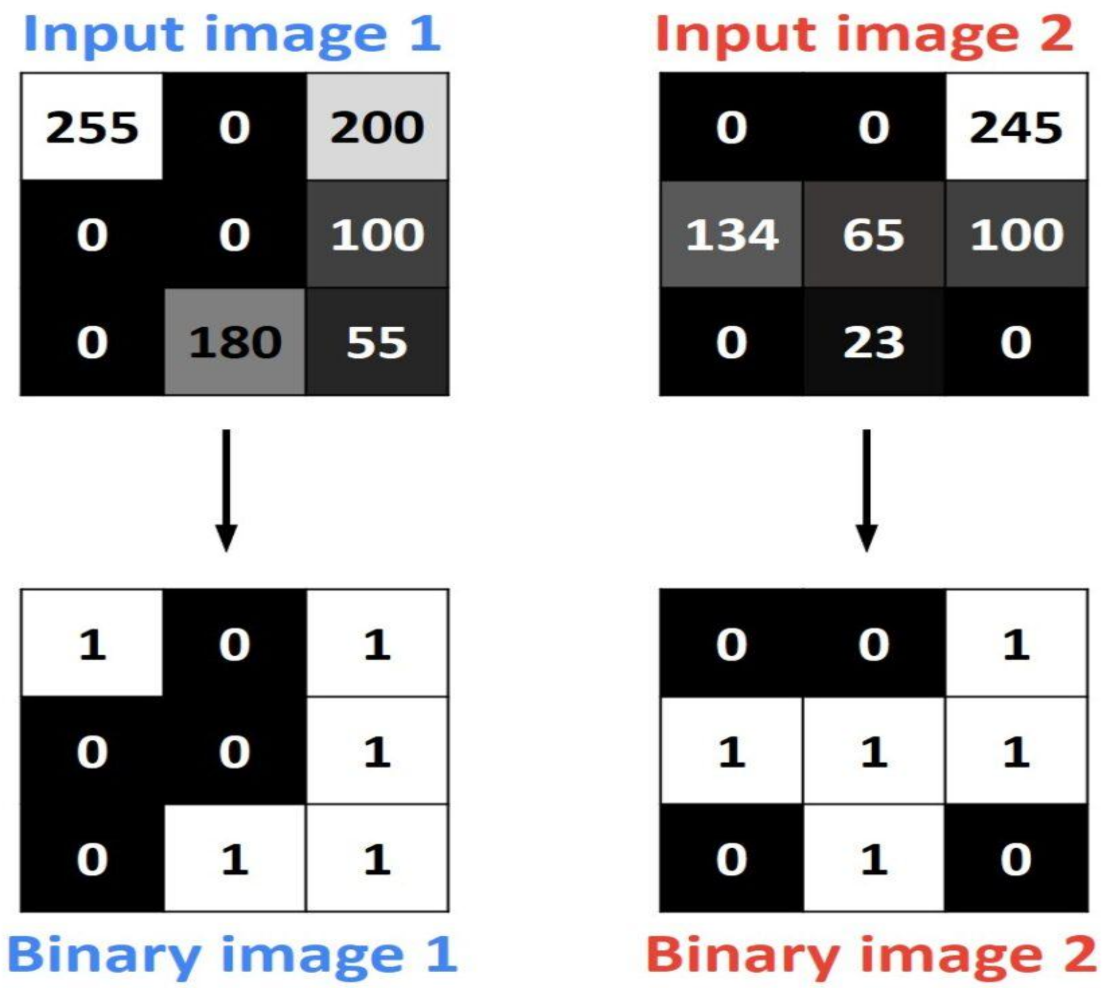
Can be used to “undo” logarithmic processing: **f(g) =** e^g

Image Logical Operations I

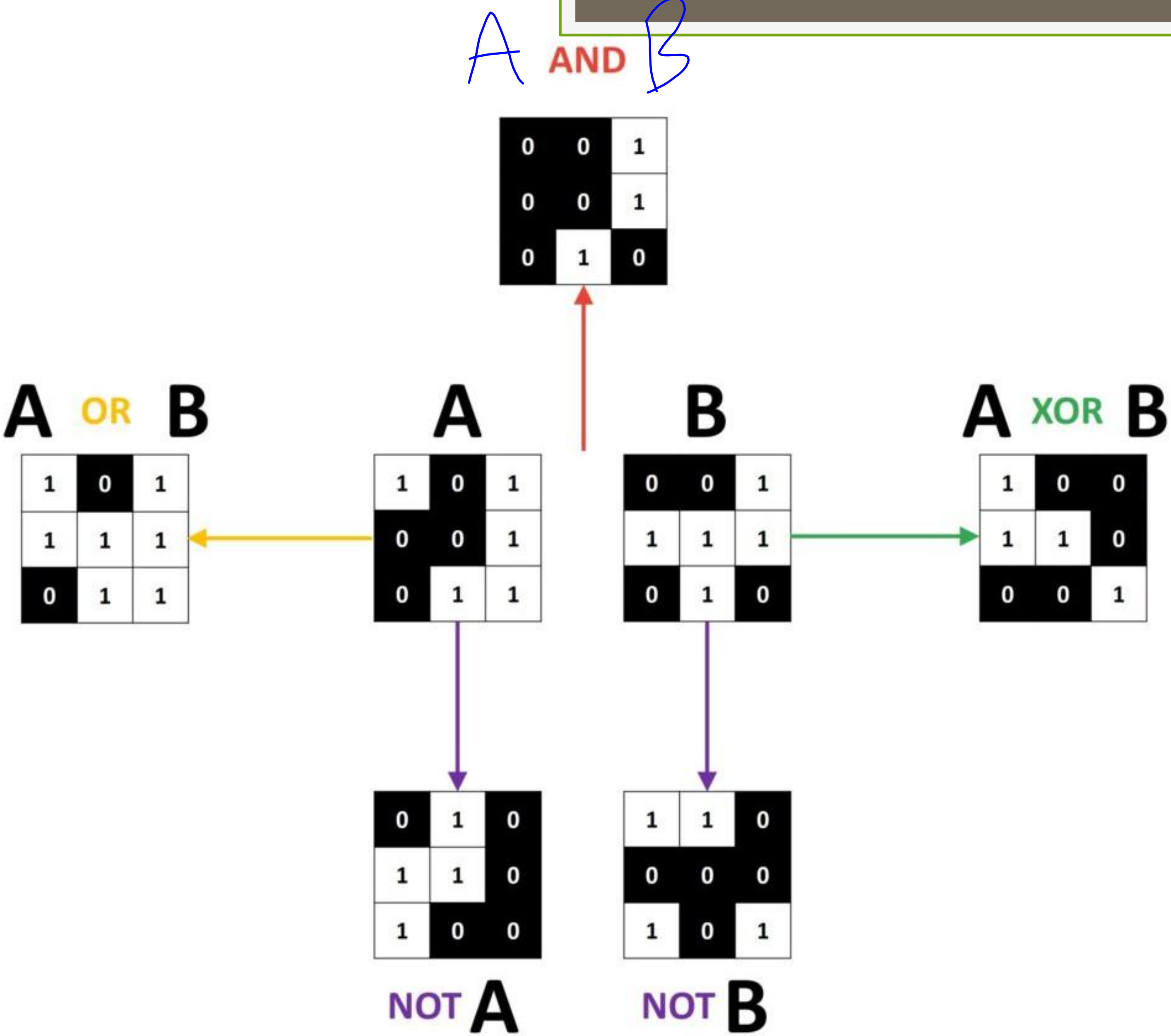
AND – True if both pixels are greater than zero
OR – True if either of the two pixels are greater than zero
XOR – True if either of the two pixels are greater than zero, but not both
NOT – Invert pixel values

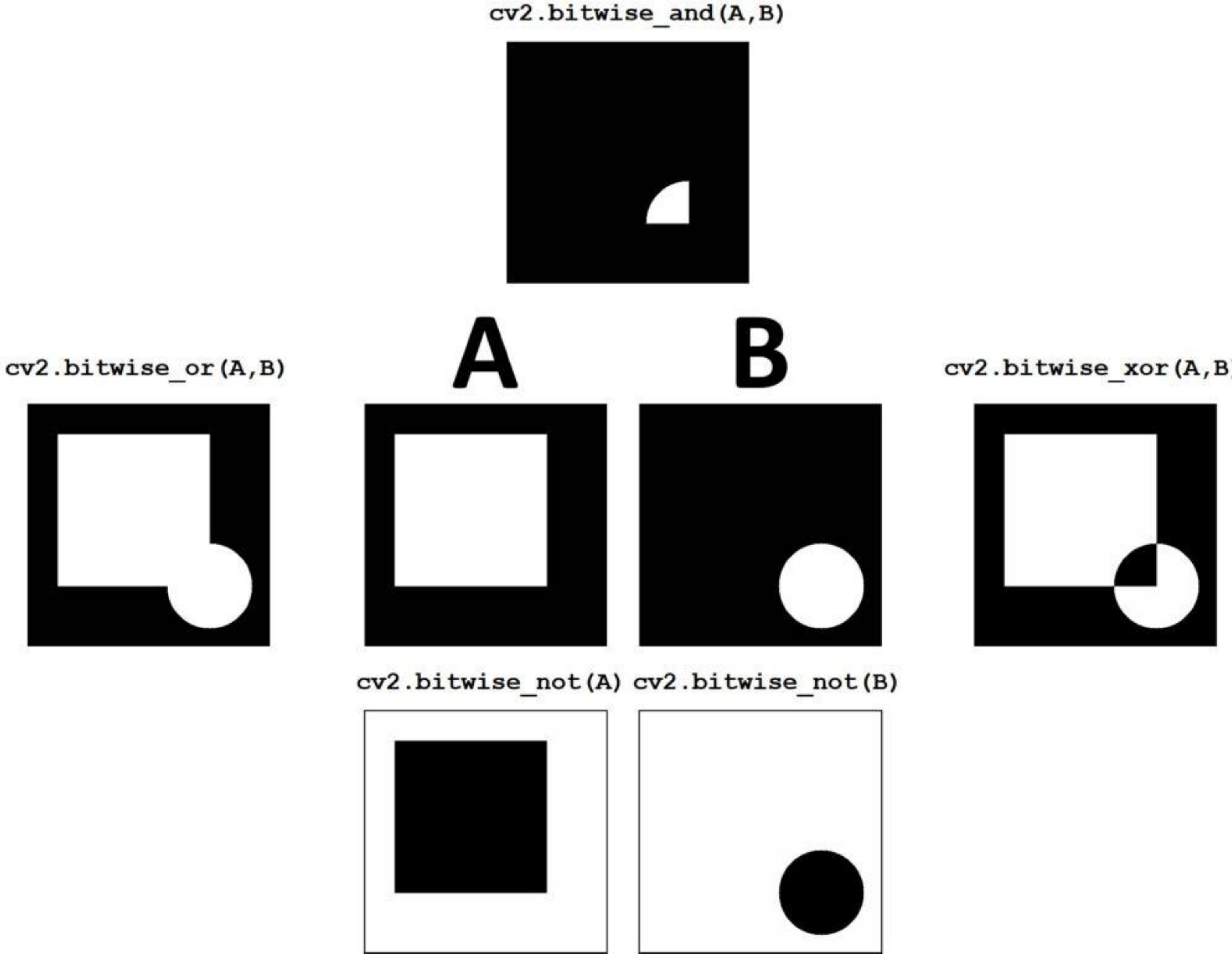
X	Y	AND	OR	XOR	NOT	
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	1
1	1	1	1	0	0	0

Example



Input image 1
255 0 200
0 0 100
0 180 55
Input image 2
0 0 245
134 65 100
0 23 0
1 0 1
0 0 1
0 1 1
Binary image 1
0 0 1
1 1 1
0 1 0
Binary image 2


A AND B
A OR B
A
B
A XOR B
NOT A
NOT B


cv2.bitwise_and(A,B)
cv2.bitwise_or(A,B)
A
B
cv2.bitwise_xor(A,B)
cv2.bitwise_not(A)
cv2.bitwise_not(B)

Discussion (Q/A)