



# Welcome to Computer Vision

# Computer Vision

Dr. Muhammad Tahir

DEPARTMENT OF COMPUTER SCIENCE,

FAST-NUCES, Peshawar

# Course Details

**LECTURES:** Tuesday
& Wednesday

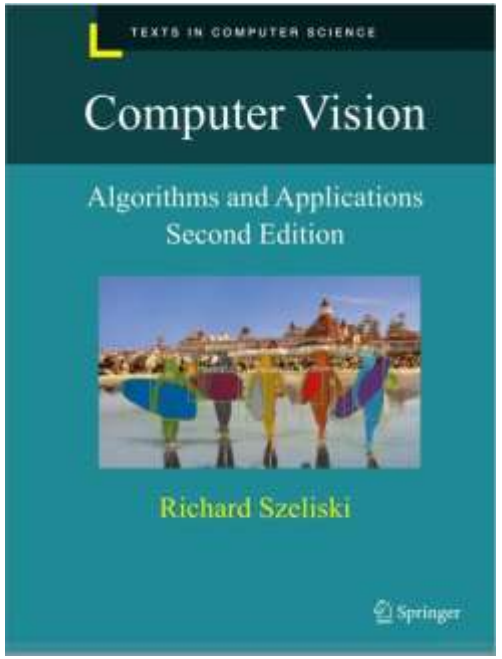
**TIMINGS:**

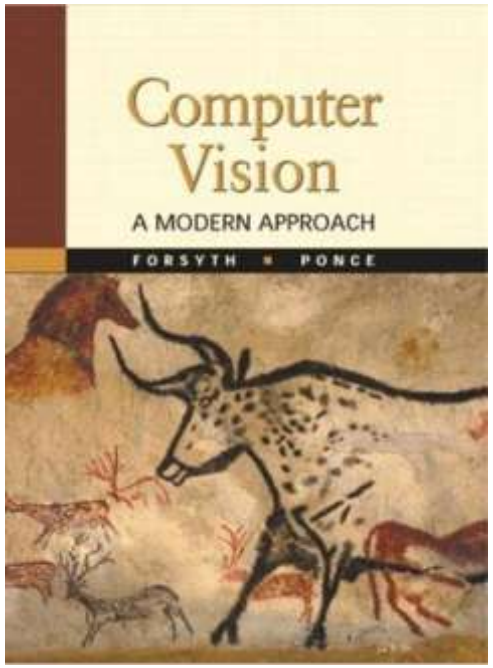
8:00 am – 9:30 am

**MY OFFICE:**

**OFFICE HOURS:**

**EMAIL: m.tahir@nu.edu.pk**



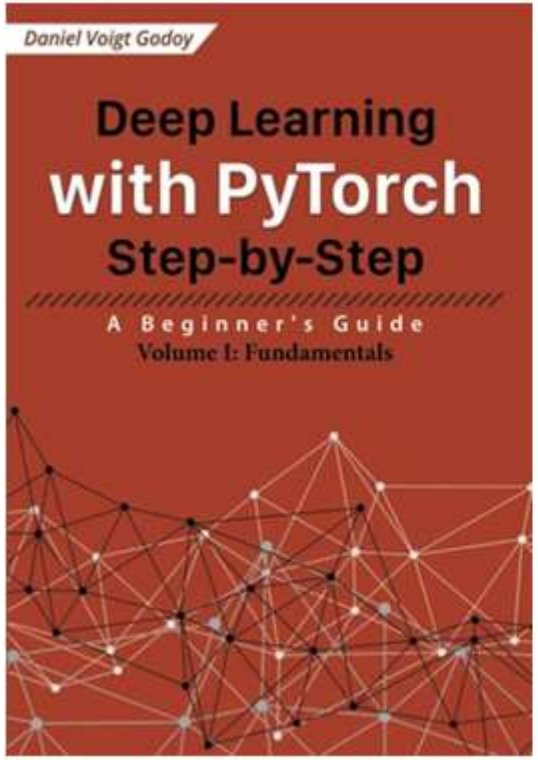

# References

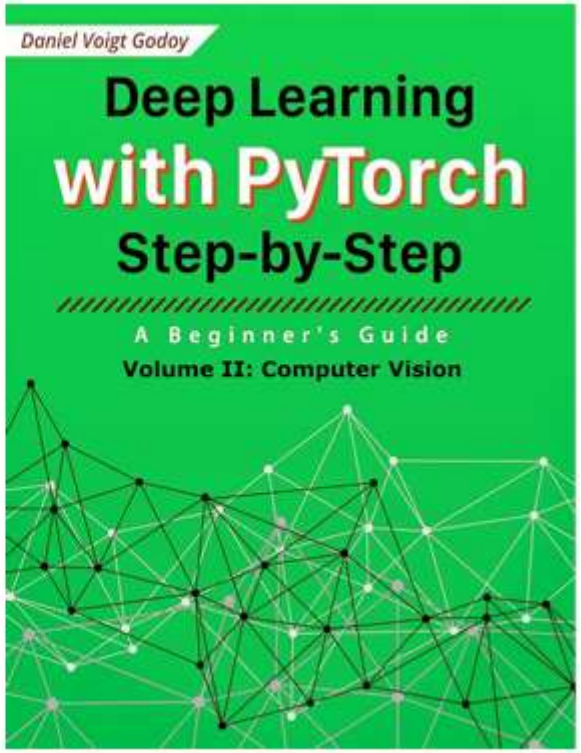
The material in these slides are based on:

1. Rick Szeliski's book: Computer Vision: Algorithms and Applications
2. Forsythe and Ponce: Computer Vision: A Modern Approach

# Recommended Books

Deep Learning with PyTorch Step-by-Step by Daniel Voigt Godoy



# Course Learning Outcomes

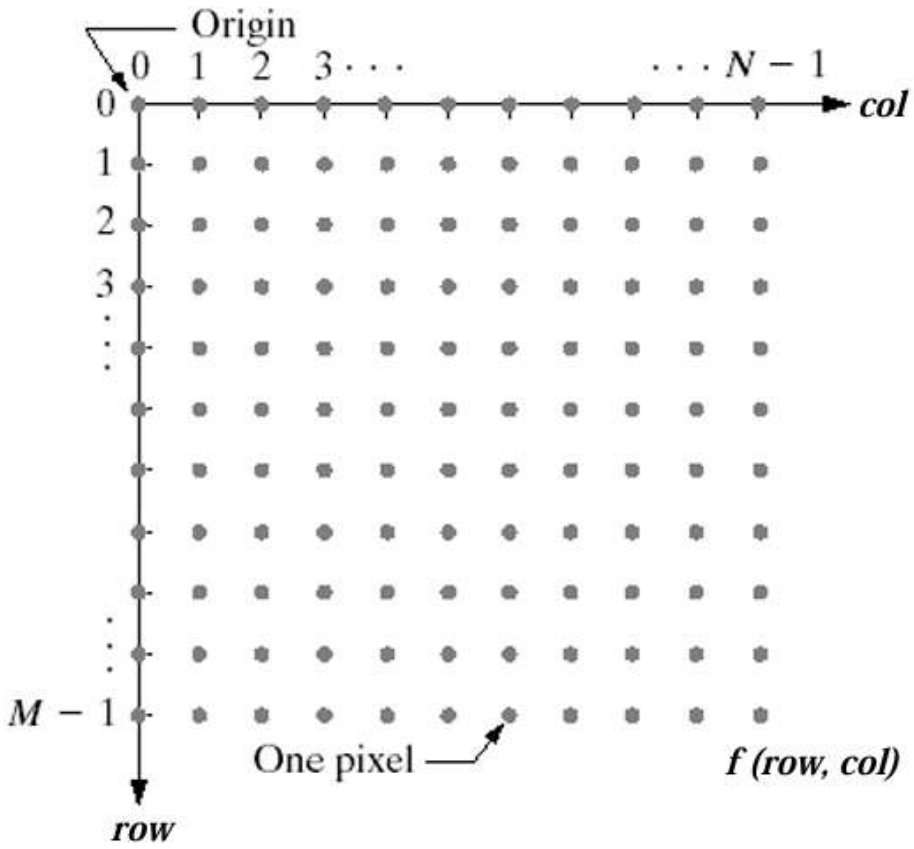
| No | CLO (Tentative) | Domain | Taxonomy Level | PLO |
|---|---|---|---|---|
| 1 | Understanding basics of Computer Vision: algorithms, tools, and techniques | Cognitive | 2 | |
| 2 | Develop solutions for image/video understanding and recognition | Cognitive | 3 | |
| 3 | Design solutions to solve practical Computer Vision problems | Cognitive | 3 | |


# Outline

- Image Representation
- Image Filtering
- Gradients
- Convolution
- Correlation

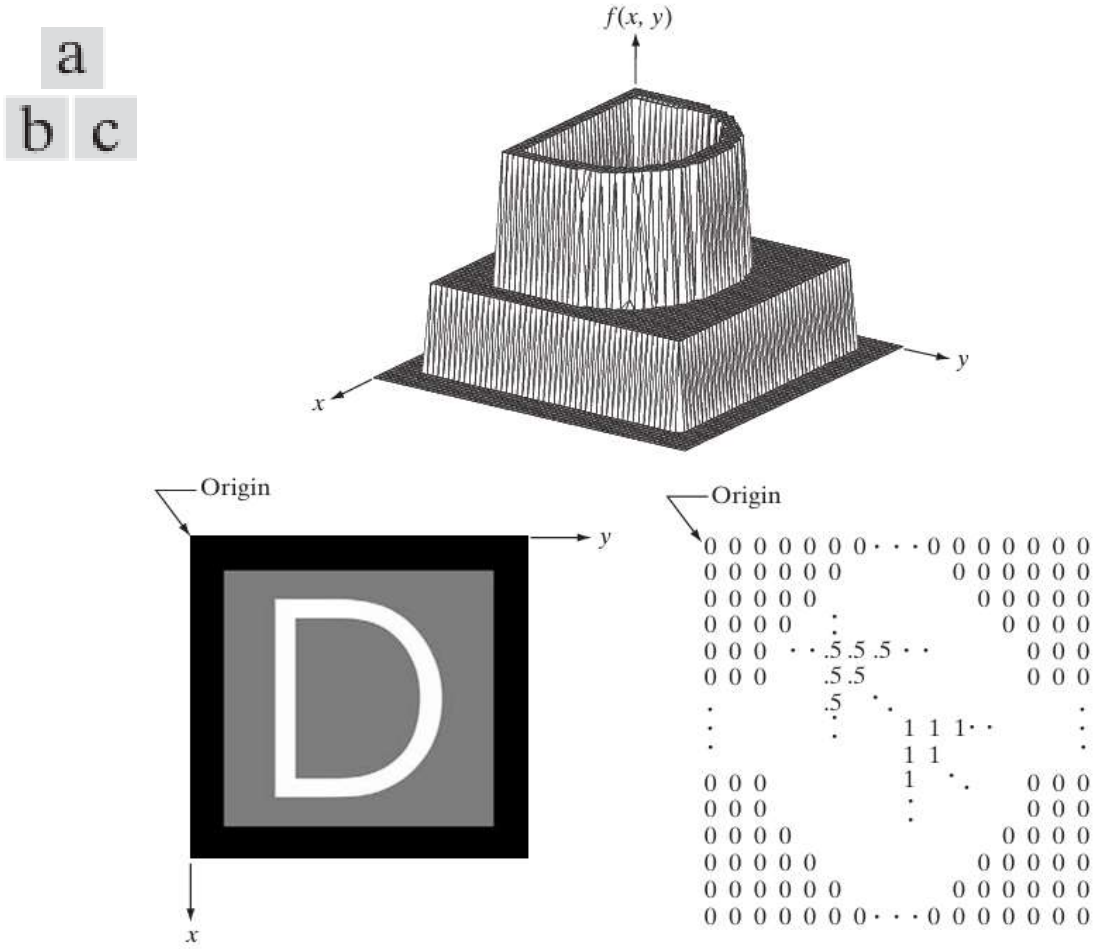
# Image Representation

- A digital image is composed of *M rows and N columns* of pixels each storing a value
- Pixel values are most often grey levels in the range 0-255(black-white)
- We will see later that images can easily be represented as matrices



# Image Representation

**FIGURE2.18** (a) Image plotted as a surface. (b) Image displayed as a visual intensity array. (c) Image shown as a 2-D numerical array (0,.5,and 1 represent black, gray, and white, respectively).

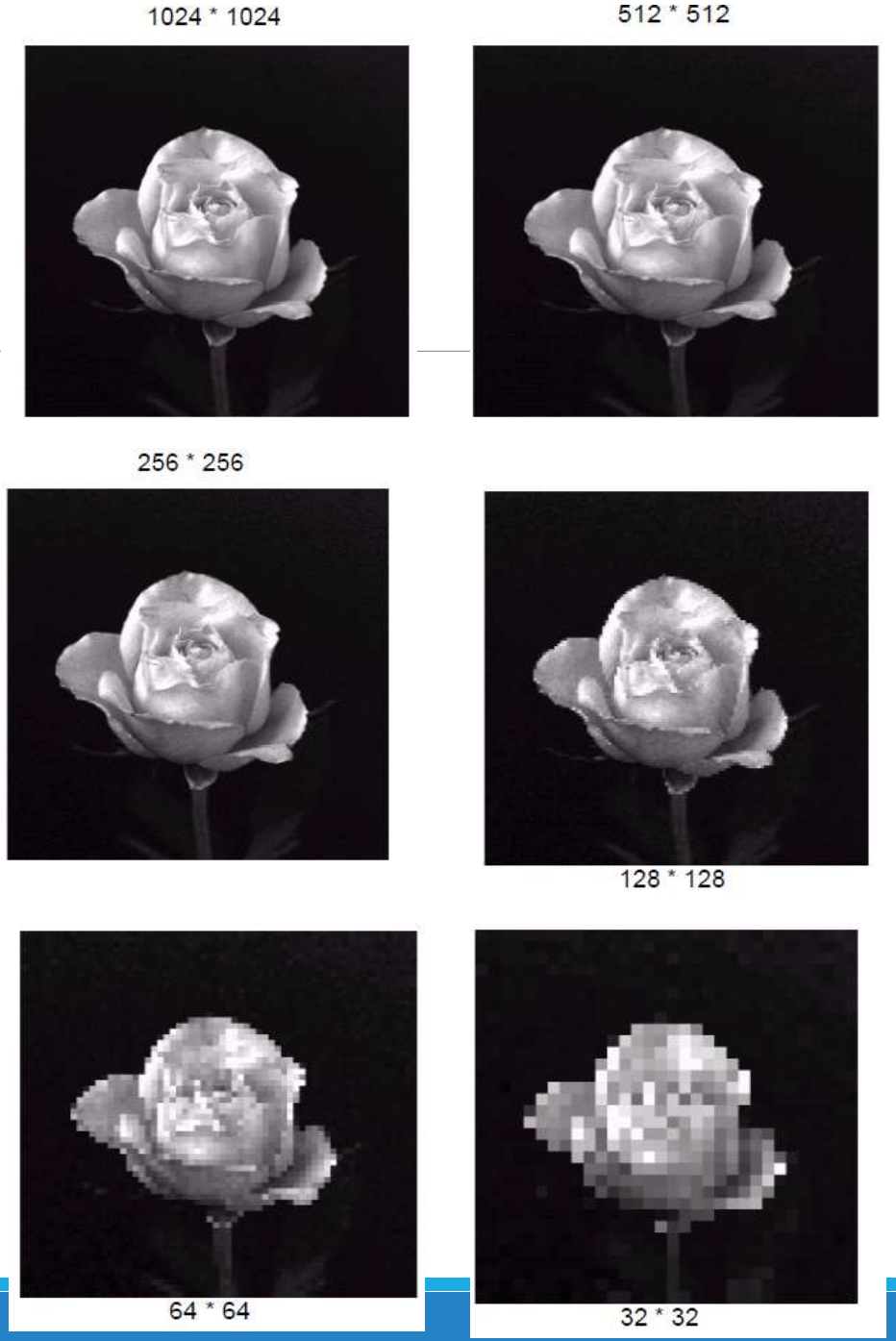


# Spatial Resolution

The spatial resolution of an image is determined by how sampling was carried out

Spatial resolution simply refers to the smallest discernible detail in an image

- Vision specialists will often talk about pixel size
- Graphic designers will talk about dots per inch (DPI)



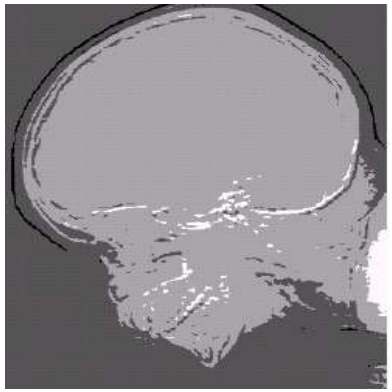
# Intensity Level Resolution

- Intensity level resolution refers to the number of intensity levels used to represent the image
- The more intensity levels used, finer the level of detail discernible in an image
- Intensity level resolution is usually given in terms of the number of bits used to store each intensity level

| Number of Bits | Number of Intensity Levels |
|---|---|
| 1 | 2 |
| 2 | 4 |
| 4 | 16 |
| 8 | 256 |
| 16 | 65,536 |

256 grey levels
(8 bits per pixel)

16 grey levels
(4 bits per pixel)



4 grey levels
(2 bits per pixel)

1 grey levels
(1 bit per pixel)

# Resolution: How much is enough?

The big question with resolution is always how much is enough?

- This all depends on what is in the image and what you would like to do with it
- Key questions include
  - Does the image look aesthetically pleasing?
  - Can you see what you need to see within the image?


The picture on the right is fine for counting the number of cars, but not for reading the number plate

# Intensity Level Resolution

•Based on the **intensity levels** and

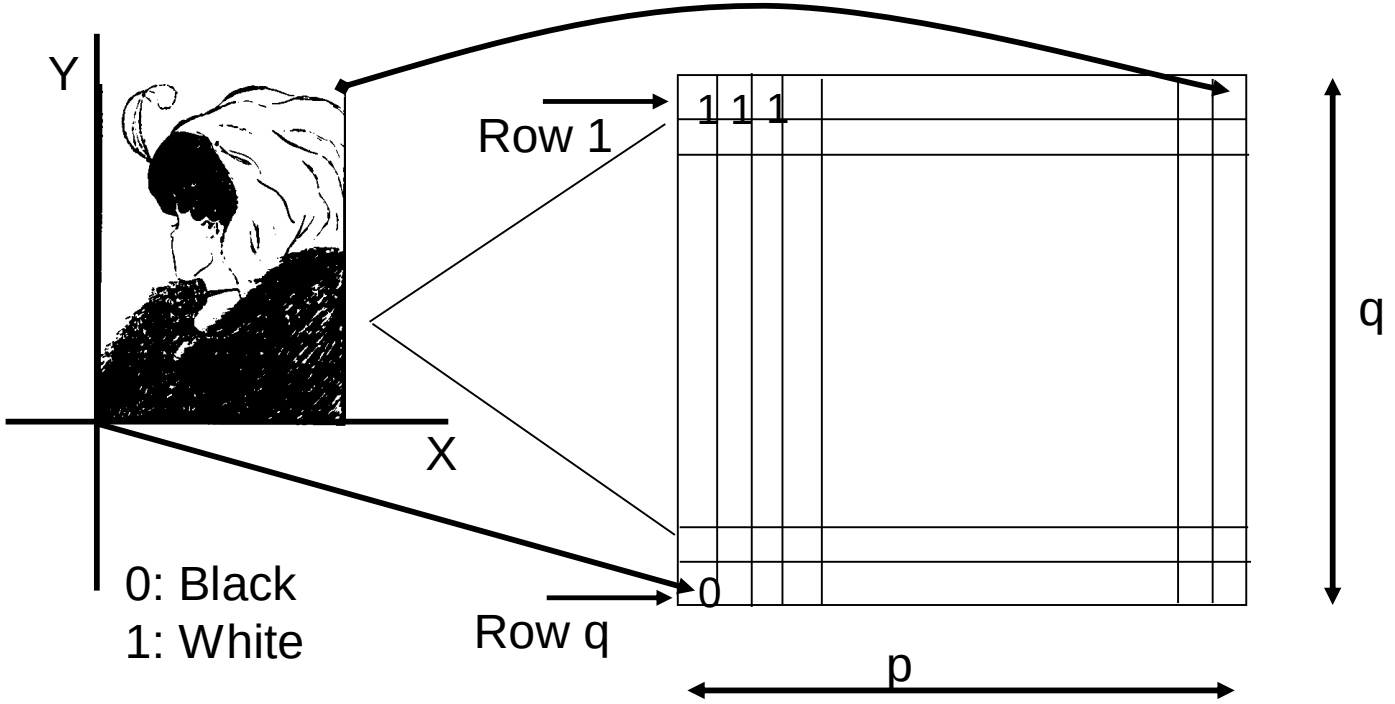
**number of channels**, we can

categorize images into:

- ✓ Binary image
- ✓ Gray-scale (or gray-tone) image
- ✓ Vector-Valued Images

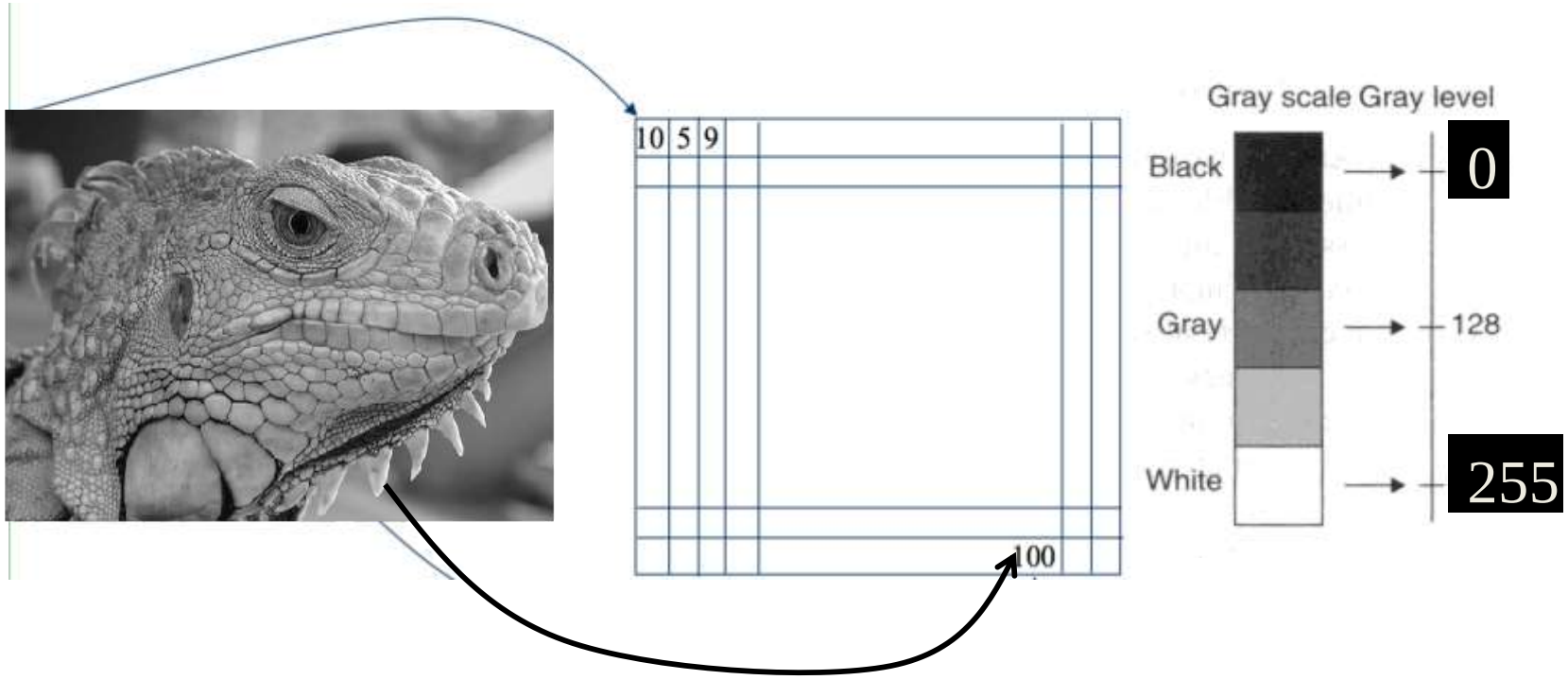
# Binary Image

A **binary image** is a digital image that has only two values (**0 or 1**) for each pixel, **0 = black** and **1 = white**



# Gray Scale Image

The intensity of each pixel is represented by a value between **<u>0 and 255</u>** (8 bits/pixel)

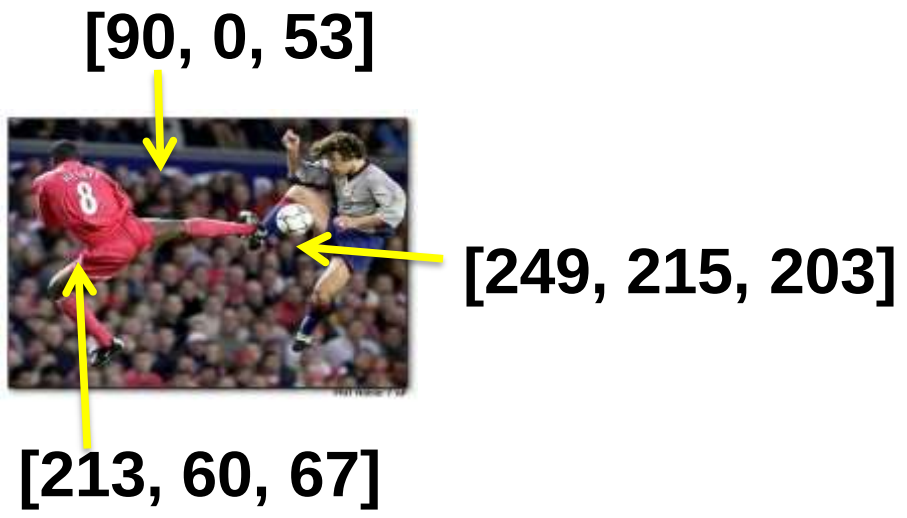


# Vector-Valued Image

•A **vector-valued image** has more than one channel or band

•Example: Color images in the RGB color model have channels for the **red**, **green**, and **blue** component.

•Each channel has a value range from 0 to 255 (like a gray-value image)



# RGB Color Images - Example

Red

Green

Blue

# Image Enhancement

- Process an image so that the **result** will be more **suitable** than the original image for a specific application.
- The suitableness is up to each **application**.
- A method which is quite useful for enhancing an image may not necessarily be the best approach for enhancing another image
- Common reasons for enhancement include
  - ✓ Improving visual quality,
  - ✓ Improving recognition accuracy,
  - ✓ Highlighting interesting detail in images
  - ✓ Removing noise from images

# Image Enhancement Methods

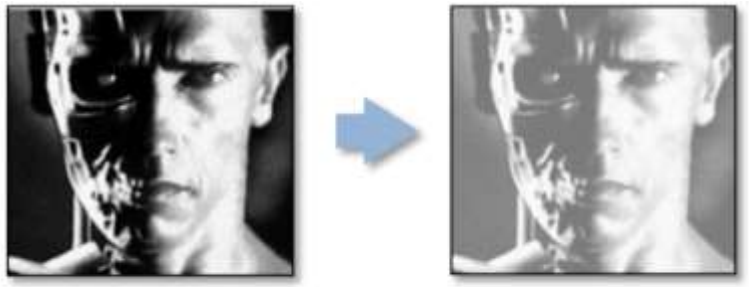
- There are two broad categories of image enhancement techniques
  - **Spatial domain Techniques:** Direct manipulation of pixel in an image (on the image plane)
  - **Frequency domain Techniques:** Processing the image based on modifying the Frequency transform of an image
- Many techniques are based on various combinations of methods from these two categories
- For the moment we will concentrate on techniques that operate in the **spatial domain**

# Intensity Transformations and Spatial Filtering

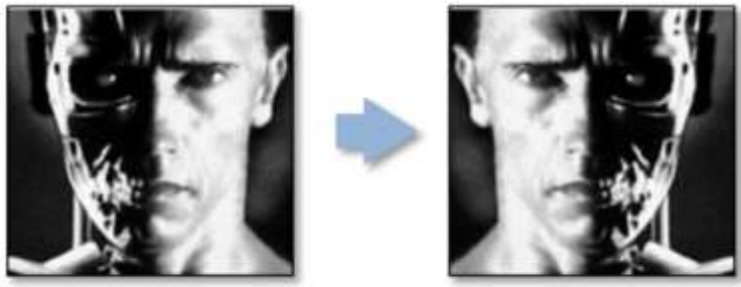
- **Spatial Domain**
  - Refers to the image plane itself that contains the pixels
- Image processing techniques in this category are based on direct manipulation of pixels in an image
- Two major categories:
  - **Intensity Transformation techniques** (point processing techniques)
    - Operate on single pixels of an image for the purpose of contrast manipulation and image thresholding
  - **Spatial Filtering techniques** (Neighborhood processing techniques)
    - Perform operations like image sharpening, image smoothing etc., by working in a neighborhood of every pixel in an image

# Point operation

- Point operation deals with pixel intensity values individually
- Enhancement at any point depends only on the image value at that point.
- The transformed output pixel value does not depend on any of the neighboring pixel value of the input image
- Point Operation Examples:
  - ✓ Image Negative
  - ✓ Contrast Stretching
  - ✓ Thresholding



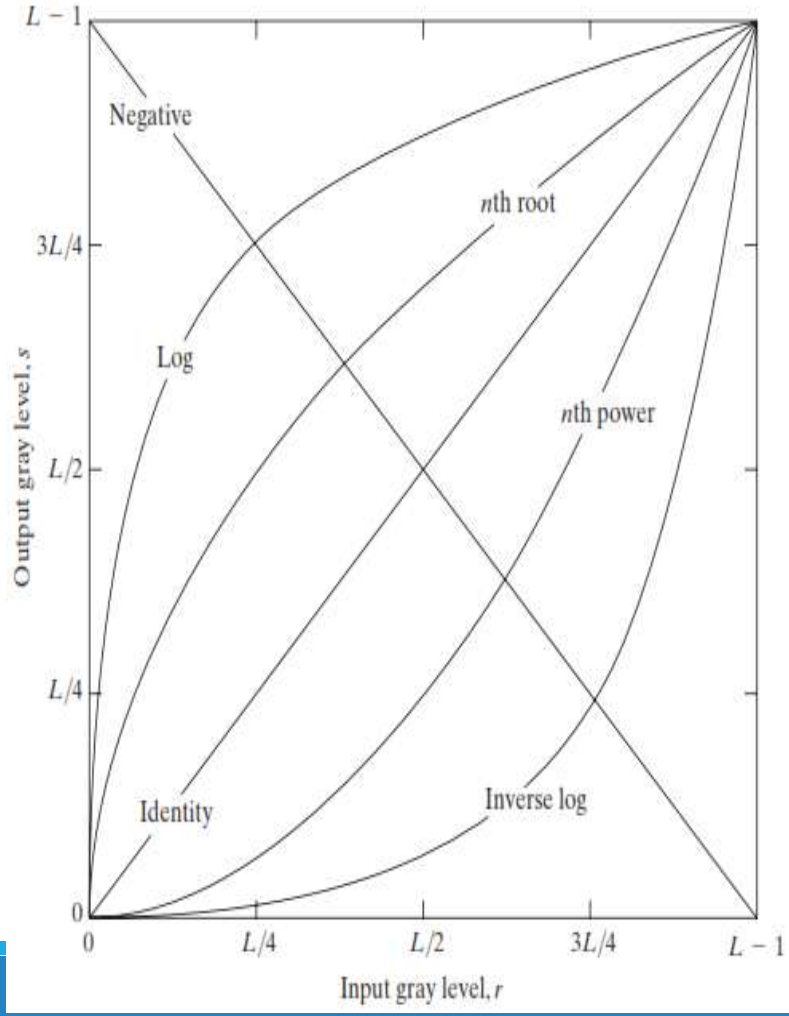
$g(x,y) = f(x,y) + 20$



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

# Point operations: Intensity Transformation

- The intensity values are altered using a particular transformation technique as per the requirement.
- Gray-level transformation function
- **S=T(r)**
  - ✓ where r is the pixels of the input image and s is the pixels of the output image.
  - ✓ T is a transformation function that maps each value of **r** to each value of **s**
- Basic types of transformation functions that are used frequently in image enhancement:
  - ✓ Linear, Logarithmic: s = c log(r + 1), Power-Law: S = C $r^{\gamma}$



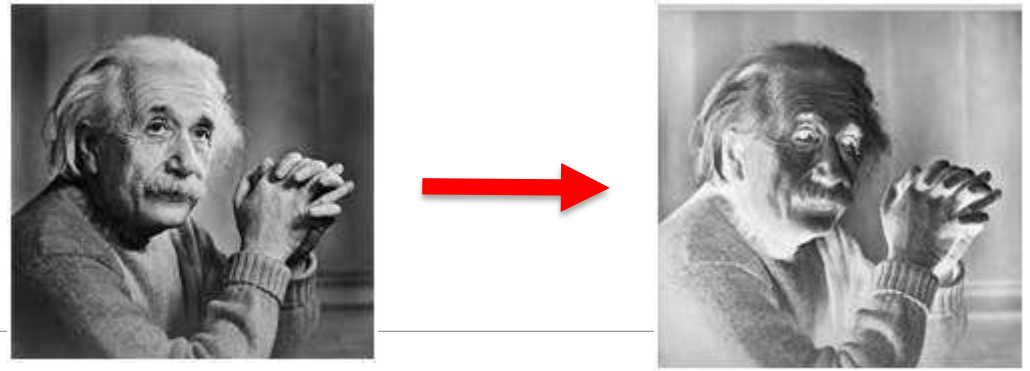
# Some Basic Intensity Transformation Functions

- Image Negatives
- Log Transformations
- Power-Law(Gamma) Transformations
- Piecewise-Linear Transformation Functions

# Image Negatives

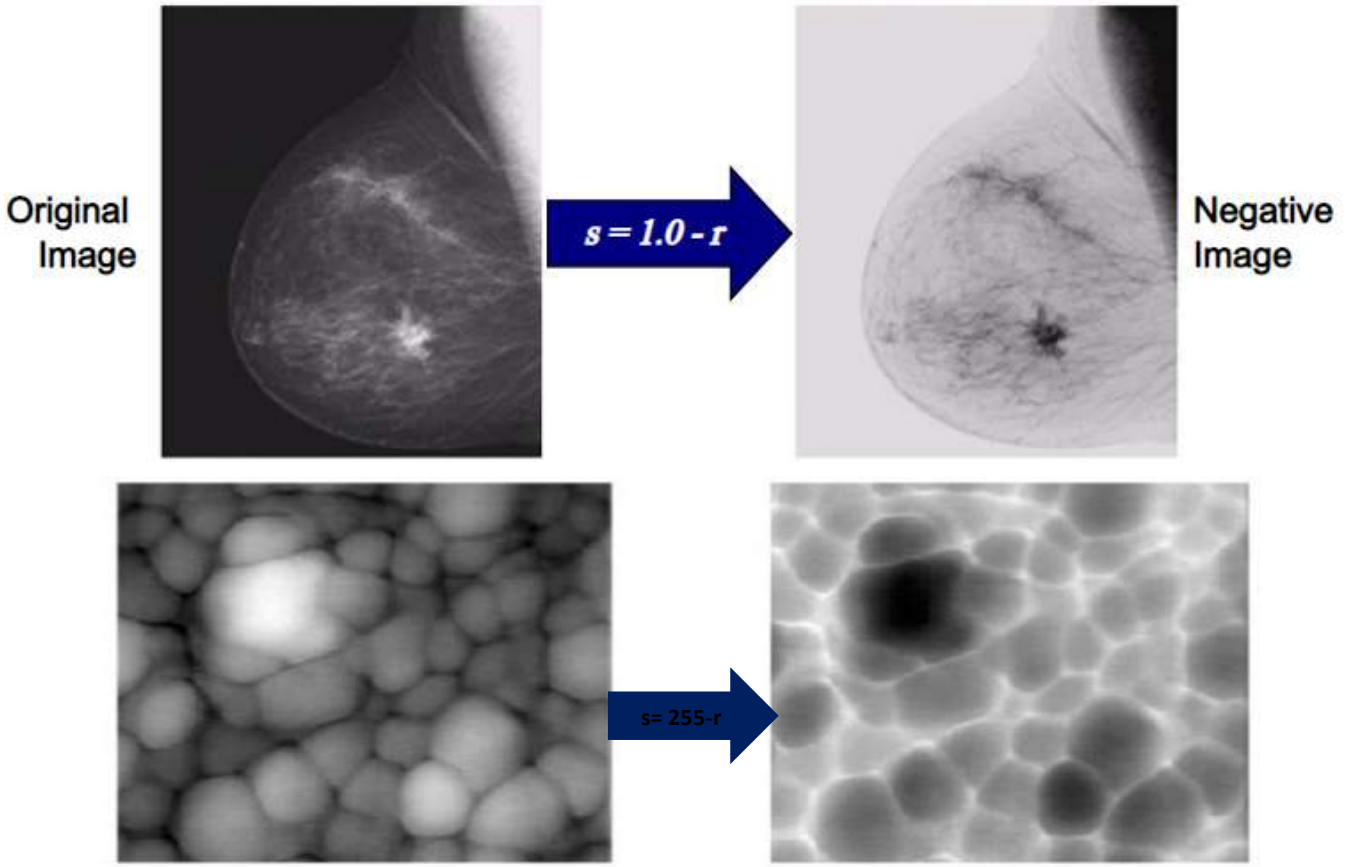
- **Reversing** the intensity levels of an image.
- **Image negative** is produced by **subtracting** each pixel from the **maximum intensity** value. e.g. for an 8-bit image, the max intensity value is $2^8 – 1 = 255$
- An image with gray level in the range [0, L-1] where L = 2n ; n = 1, 2…
- Negative transformation : $s = L – 1 – r$

# Image Negatives



- Reversing the intensity levels of an image.
- Image negative is produced by subtracting each pixel from the maximum intensity value. e.g. for an 8-bit image, the max intensity value is $2^8 – 1 = 255$
- An image with gray level in the range [0, L-1] where L = 2n ; n = 1, 2…
- Negative transformation : $s = L – 1 – r$

# Image Negatives: Applications



# Image Negatives: Activity

An Image with n = 7

| | | |
|---|---|---|
| 95 | 22 | 36 |
| 50 | 90 | 6 |
| 84 | 5 | 13 |

Image Negative

# Image Negatives: Activity

An Image with n = 7

| | | |
|---|---|---|
| 95 | 22 | 36 |
| 50 | 90 | 6 |
| 84 | 5 | 13 |

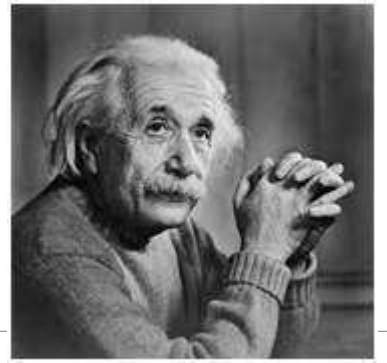
Image Negative

| | | |
|---|---|---|
| 32 | 105 | 91 |
| 77 | 37 | 121 |
| 43 | 122 | 114 |

# Some Basic Intensity Transformation Functions

- Image Negatives
- Log Transformations
- Power-Law(Gamma) Transformations
- Piecewise-Linear Transformation Functions

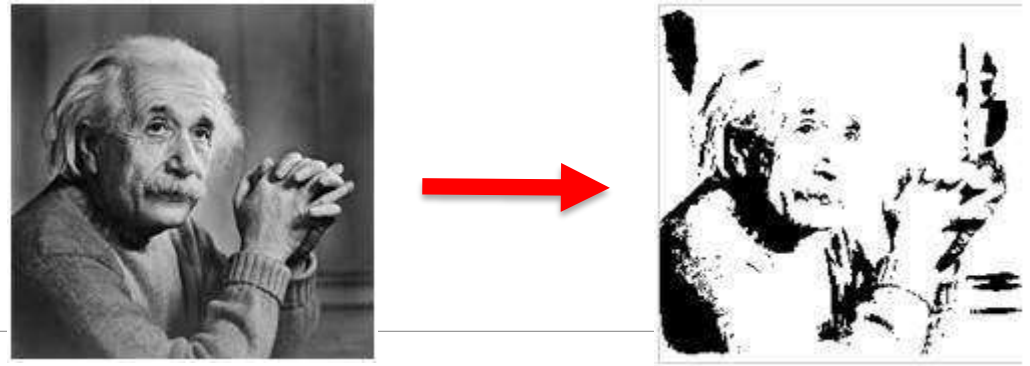
# Log Transformations



- Log curve maps a **narrow range** of low gray-level values in the input image into a **wider range** of output levels.
- c is a constant and $r \geq 0$
- Used to **expand** the values of **dark pixels** in an image while **compressing** the **higher-level values**.

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

# Log Transformations



- Log curve maps a **narrow range** of low gray-level values in the input image into a **wider range** of output levels.

- c is a constant and $r \geq 0$

- Used to **expand** the values of **dark pixels** in an image while **compressing** the **higher-level values**.

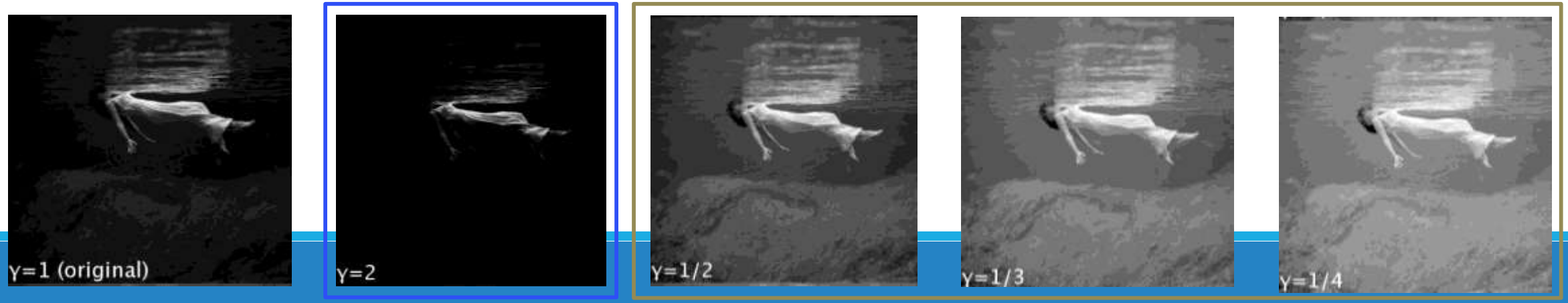
$$s = c \log(r + 1)$$

# Some Basic Intensity Transformation Functions

- Image Negatives
- Log Transformations
- Power-Law (Gamma) Transformations
- Piecewise-Linear Transformation Functions

# Power-Law (Gamma) Transformations

- Gamma correction function is used to correct image's luminance.
- Power-law transformations: $s=cr^{\gamma}$ or $s=c(r+\varepsilon)^{\gamma}$
  - ✓ $\gamma$ : gamma, gamma correction
  - ✓ $\gamma$ <1 maps a **narrow range of dark** input values into a **wider range** of output values [**increase brightness**]
  - ✓ $\gamma$ >1 maps a **narrow range of bright** input values into a **wider range** of output values [**increase darkness**]



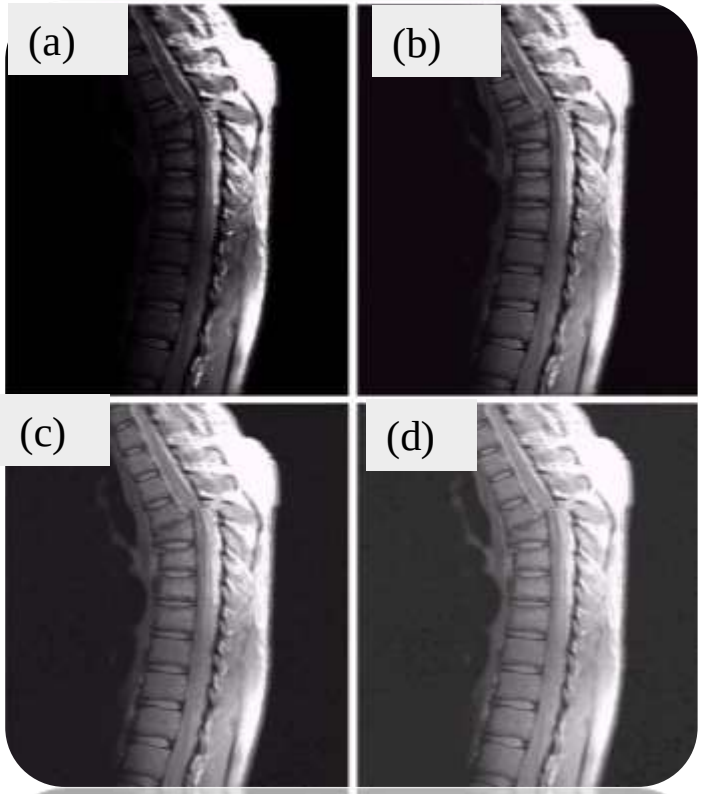
# Power-Law (Gamma) Transformations

a) A magnetic resonance image (MRI) of an upper thoracic human spine with a fracture dislocation and spinal cord impingement
   - The picture is predominately dark
   - An expansion of gray levels are desirable ⇨ needs $\gamma < 1$

b) result after power-law transformation with $\gamma = 0.6$, c=1

c) transformation with $\gamma = 0.4$ (best result)

d) transformation with $\gamma = 0.3$ (under acceptable level)



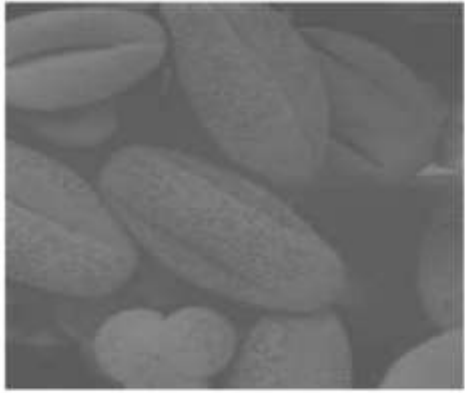
# Some Basic Intensity Transformation Functions

- Image Negatives
- Log Transformations
- Power-Law (Gamma) Transformations
- Piecewise-Linear Transformation Functions

# Piecewise-Linear Transformation Functions

- Some commonly used piece-wise linear transformations are:
  - ✓ Contrast Stretching
  - ✓ Gray-level Slicing
  - ✓ Clipping

# Contrast Stretching



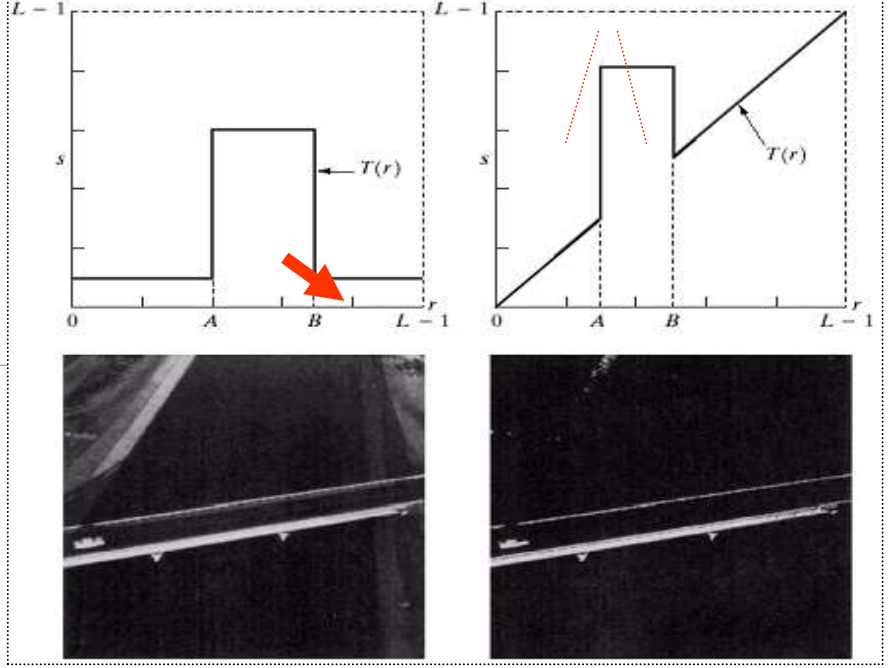
Original Image

Contrast Enhanced Image

- It is the difference between the intensity values of darker and brighter pixels
- Contrast stretching expands the range of intensity levels in an image.
- Contrast stretching is done in three ways:
  1. Multiplying each input pixel intensity value with a constant scalar.
     - Example: s=2*r
  2. Using Histogram Equivalent
  3. Applying a transform which makes dark portion darker by assigning slope of < 1 and bright portion brighter by assigning slope of > 1.

# Gray-level Slicing

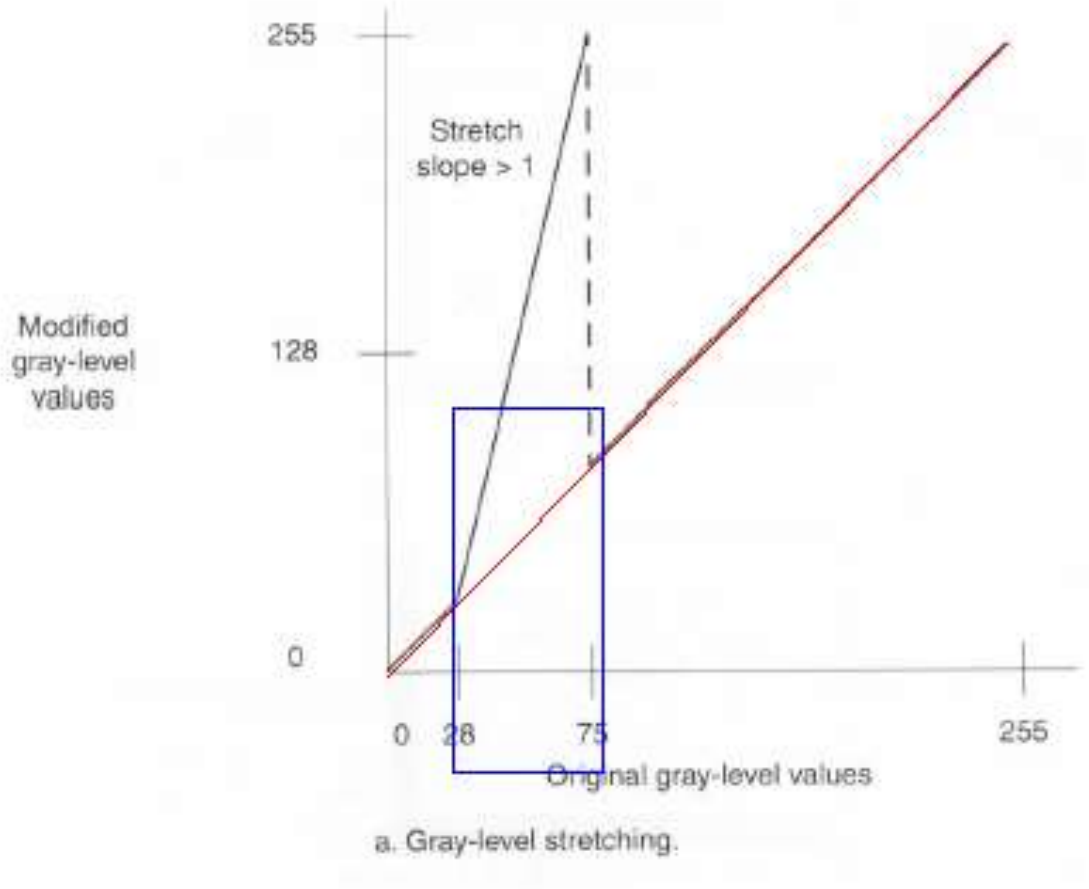
- Highlighting a specific range of gray levels in an image
- Display a high value of all gray levels in the range of interest and a low value for all other gray levels
  a) transformation highlights range [A,B] of gray level and reduces all others to a constant level
  b) transformation highlights range [A,B] but preserves all other levels




(a) (b)

# Gray-level Slicing Example

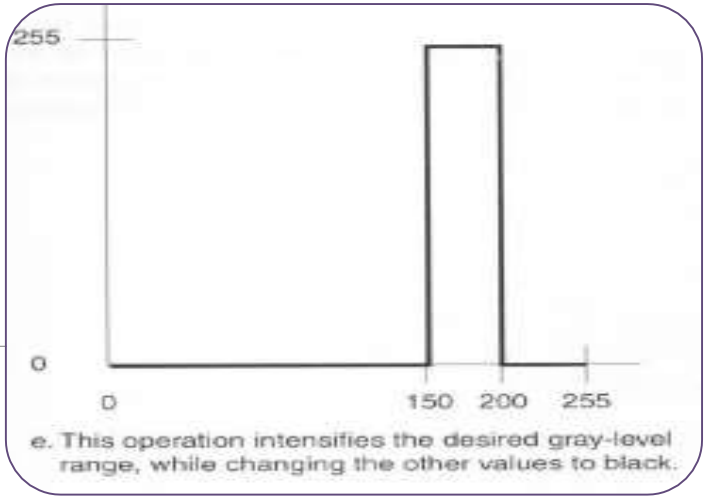
b. Original image.



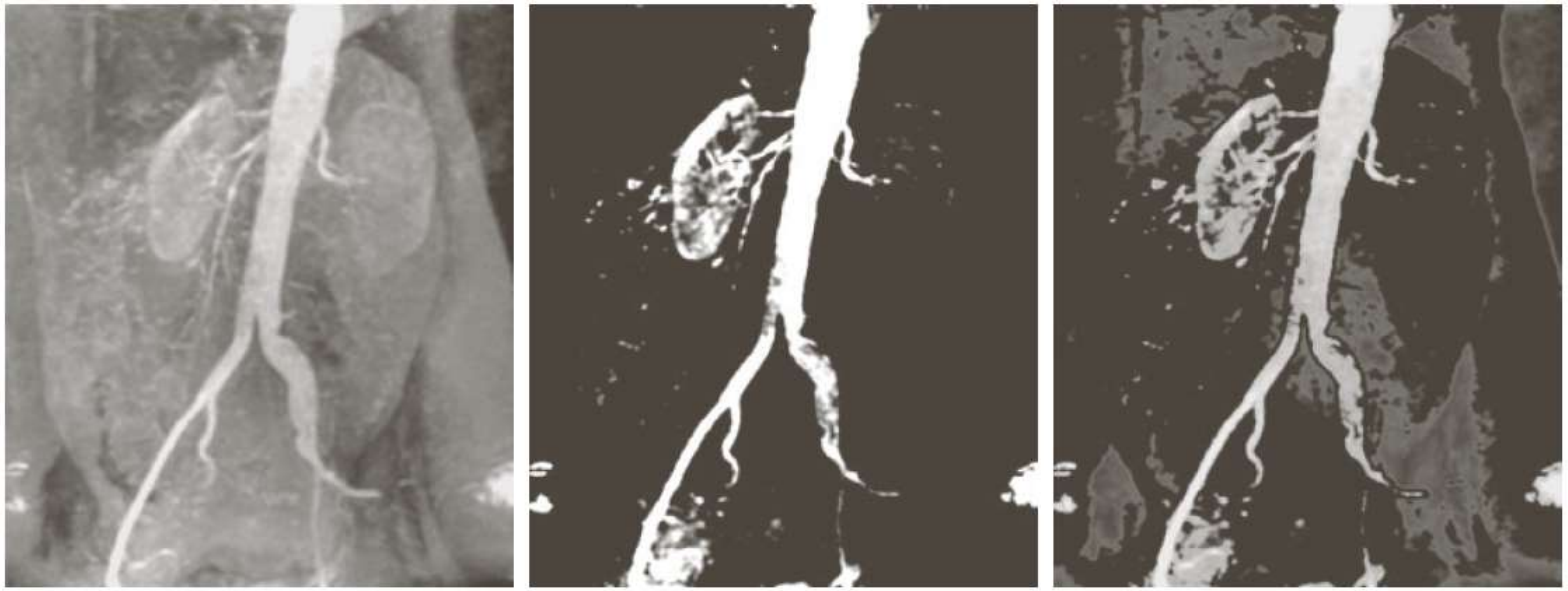

a. Gray-level stretching.

c. Image after modification.

# Gray-level Slicing Example

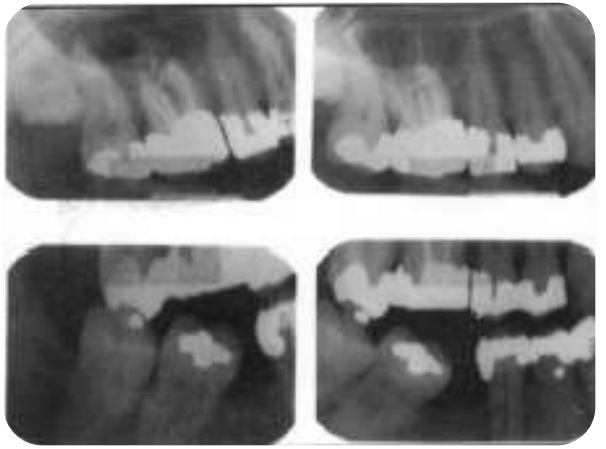


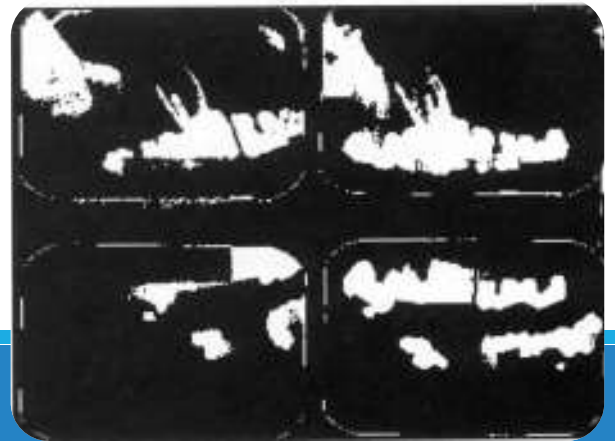

e. This operation intensifies the desired gray-level range, while changing the other values to black.



a b c

**FIGURE 3.12** (a) Aortic angiogram. (b) Result of using a slicing transformation of the type illustrated in Fig. 3.11(a), with the range of intensities of interest selected in the upper end of the gray scale. (c) Result of using the transformation in Fig. 3.11(b), with the selected area set to black, so that grays in the area of the blood vessels and kidneys were preserved. (Original image courtesy of Dr. Thomas R. Gest, University of Michigan Medical School.)



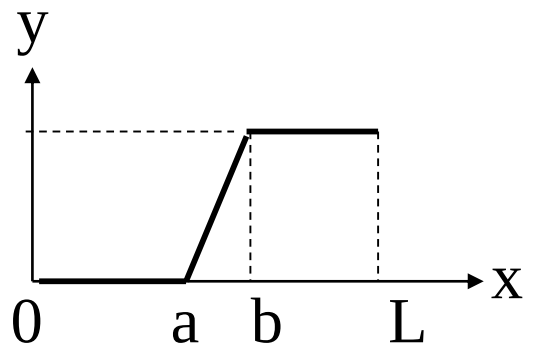


# Clipping

- This function truncates all intensities below the optional value Min to Min and all intensities above the optional value Max to Max.

- Clipping can be used to remove **unwanted features**, **noise**, or **extraneous information** from an image.

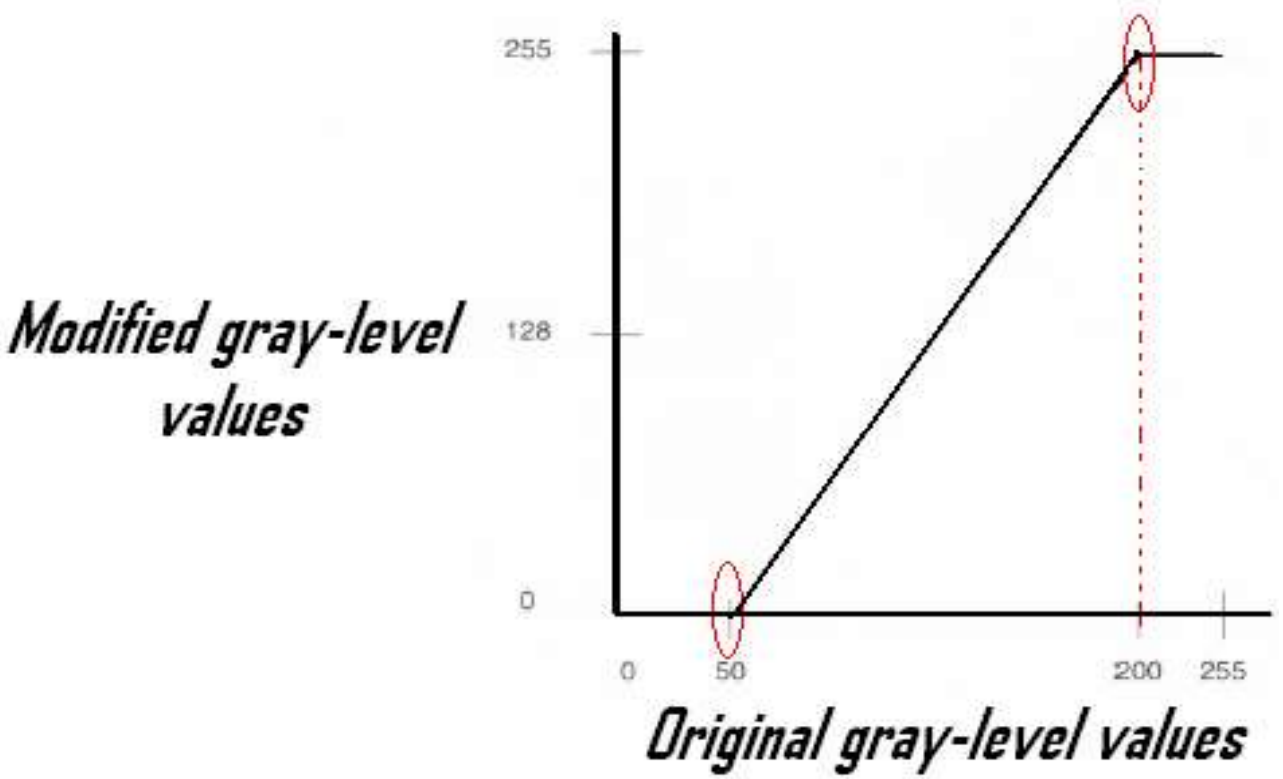
$$y = \begin{cases} 0 & 0 \le x < a \\ \beta(x-a) & a \le x < b \\ \beta(b-a) & b \le x < L \end{cases}$$




$a = 50, b = 150, \beta = 2$

# Clipping Example

e. Original image.




f. Image after modification.

# Piecewise-Linear Transformation Functions

- **Advantage:**
  - ✓ The form of piecewise functions can be arbitrarily complex.
  - ✓ A Practical Implementation of some important transformations can be formulated only as piecewise functions.

- **Disadvantage:**
  - ✓ Their specification requires considerably more user input.

# Filter (Mask) Operations

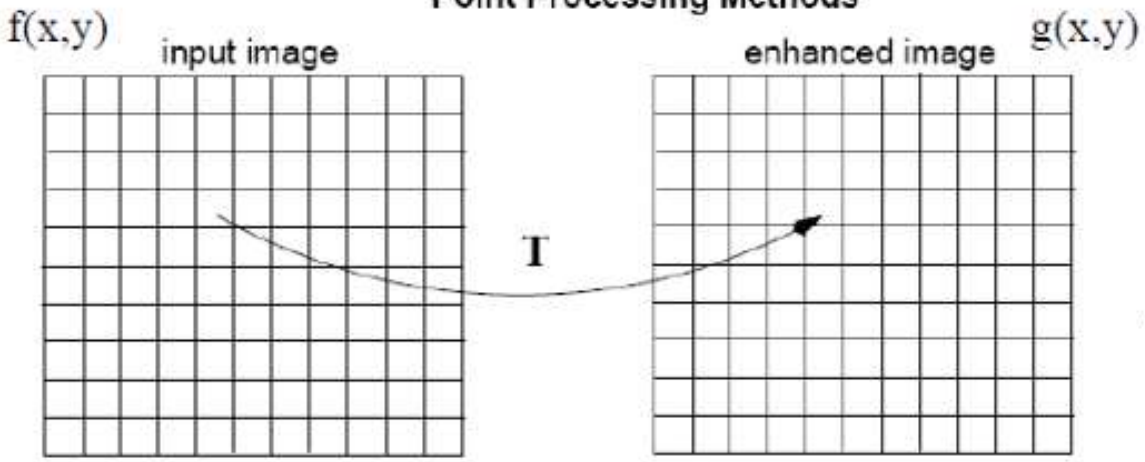
# Why Spatial Domain?

- Some image processing tasks are easier, or more meaningful to implement in the spatial domain, while others are best suited for other domains.
- Spatial domain techniques are more efficient computationally and require fewer processing resources to implement.
- Point-operations don’t know anything about their neighbors
- Most image features (edges, textures, etc) involve a spatial neighborhood of pixels
- If we want to enhance or manipulate these features, we need to go beyond point operations

# Neighborhood Processing

- To modify or change a pixel value we only need to know its local neighbour(s) gray values.
- Also referred as local processing
- Spatial Filtering is a special case of neighborhood processing

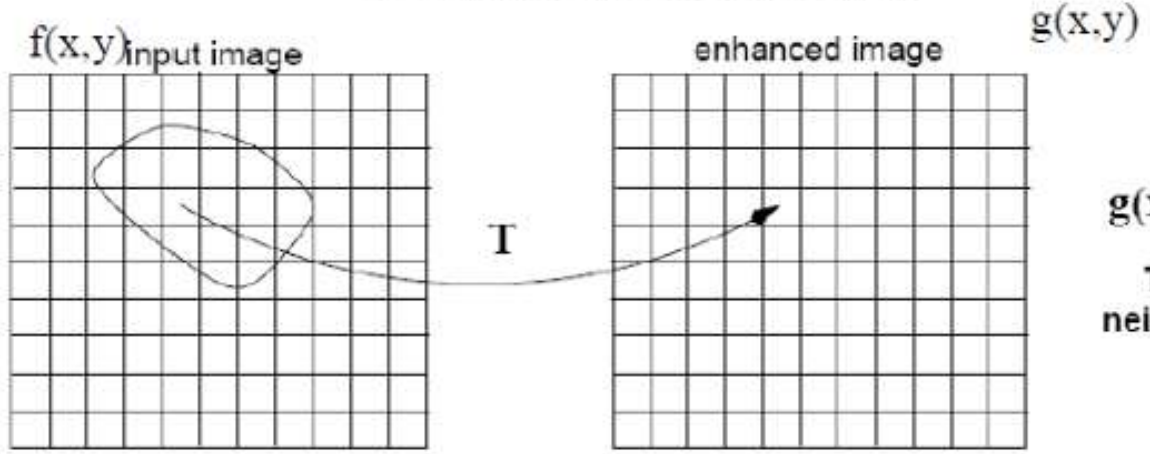
## Point Processing Methods




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

T operates on 1 pixel

## Area or Mask Processing Methods



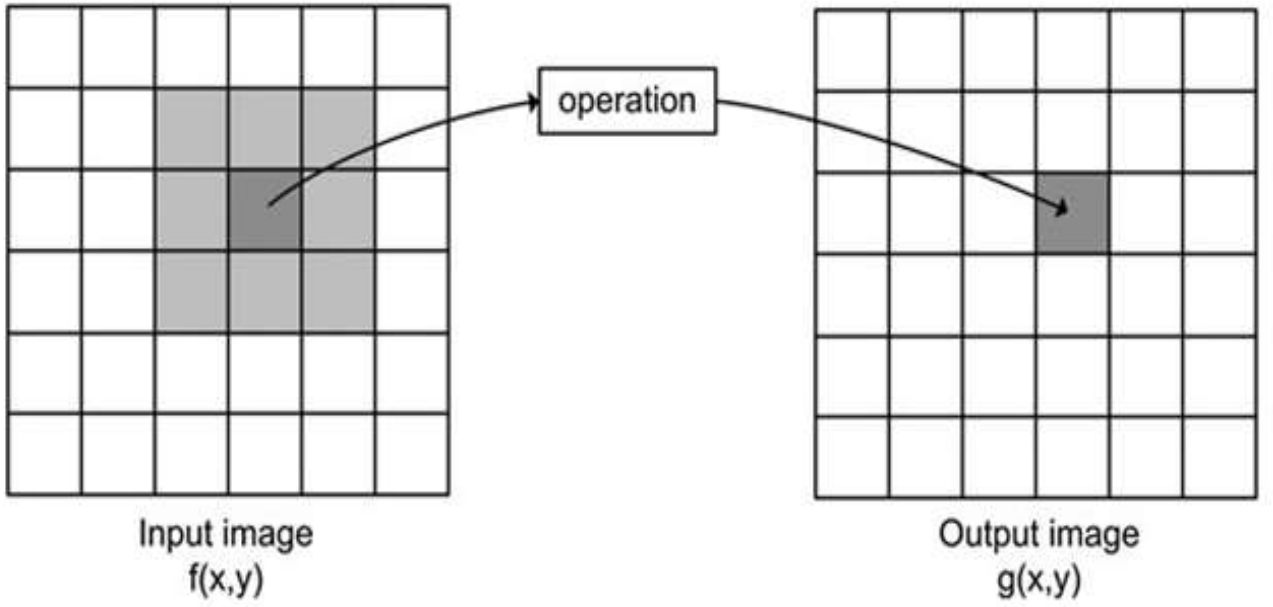

$$g(x,y) = T[f(x,y)]$$

T operates on a neighborhood of pixels

# Spatial Filtering

- Spatial domain filtering

- Capabilities of point operations are limited
- Filters: combine pixel's value + values of neighbors



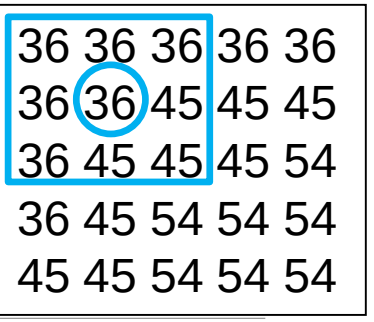
# What Point Operations Can’t Do

- Combining multiple pixels needed for certain operations:
  - **Enhance an image**, e.g., denoise, Blurring, Sharpening.
  - **Extract information**, e.g., texture, edges.
  - **Detect patterns**, e.g., template matching.

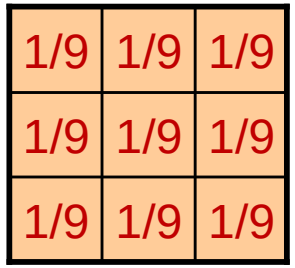
→

# Filtering Operations Use “filters”

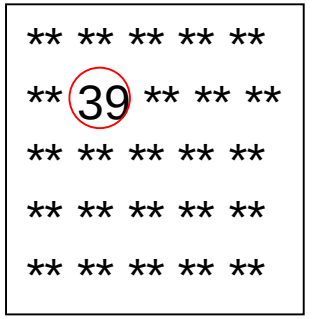
- **Filters (masks)** operate on a neighborhood of pixels.
- The sub-image is called a **filter** (or mask, kernel, template, window).
- The values in a filter sub-image are referred to as **coefficients**, rather than pixels.
  - ✓ A mask of coefficients is centered on a pixel.
  - ✓ The mask **coefficients** are **multiplied** by the pixel values in its neighborhood, and the products are **summed**.
  - ✓ The result goes into the corresponding pixel position in the output image.




Input Image




filter




Output Image

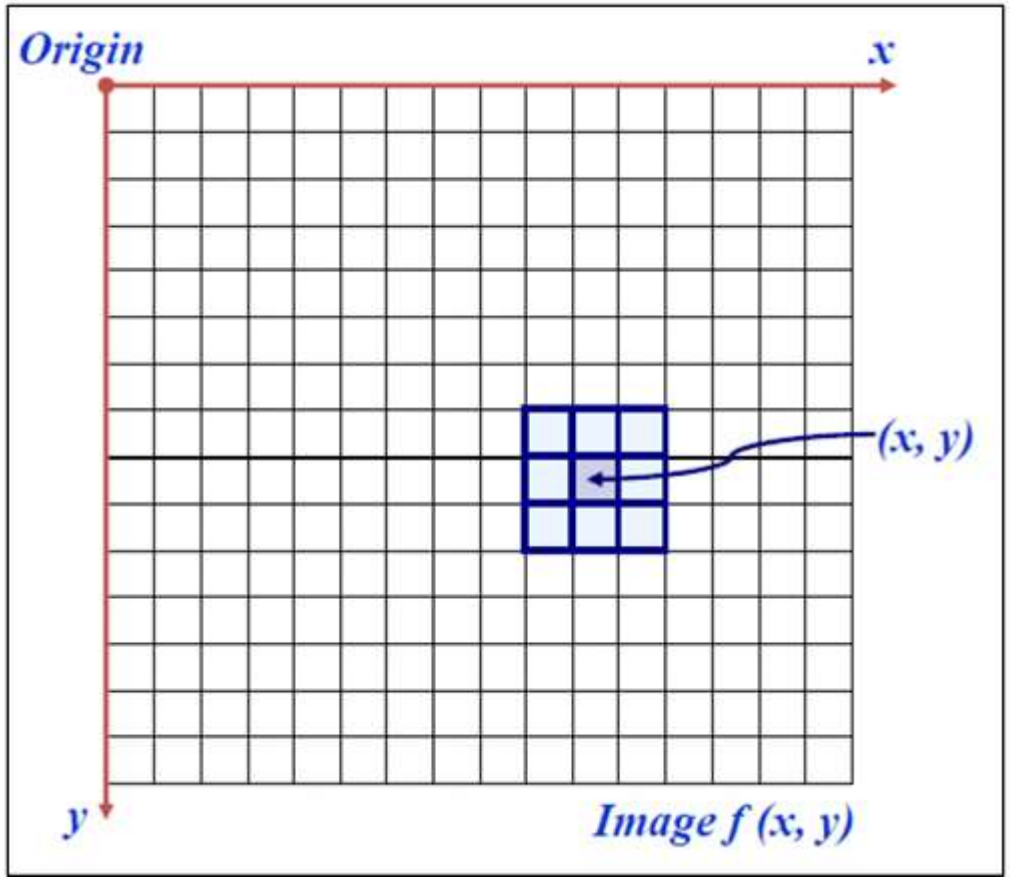
# Simple Neighbourhood Operations

- Some simple neighborhood operations include:
  - ✓ **Min:** Set the pixel value to the **minimum** in the neighborhood
  - ✓ **Max:** Set the pixel value to the **maximum** in the neighborhood
  - ✓ **Average:** Set the pixel value to the **average** of the neighborhoods
  - ✓ **Median:** The median value of a set of numbers is the **midpoint** value in that set (e.g., from the set [1, 7, 15, 18, 24] **15** is the median). Sometimes the median works better than the average

# Spatial Filtering

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

- Move a "mask" (e.g. a square) over the complete image pixel-by-pixel to create a new image
- In this new image, each pixel gray level value is written as the linear combination of the neighbor pixels.
- "Mask" defines the coefficients of linear combination.
  - Mask is also called Filter or Template



# Filters

Smoothing Spatial Filters

For blurring and noise reduction

- Smoothing Linear Filters
  - Averaging filter or Low pass filter
- Order Statistic (Nonlinear) Filters
  - Response is based on ordering or ranking of pixels under the kernel
  - Median Filter
    - Impulse noise (salt and pepper noise)
  - Max Filter
  - Min Filter

**FIGURE 3.31** Another representation of a general $3 \times 3$ filter mask.

| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |
| $w_4$ | $w_5$ | $w_6$ |
| $w_7$ | $w_8$ | $w_9$ |

# Averaging Filters

$\frac{1}{9} \times$

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

Standard average

$\frac{1}{16} \times$

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

Weighted average

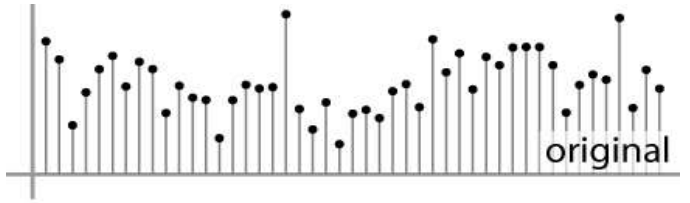
a b

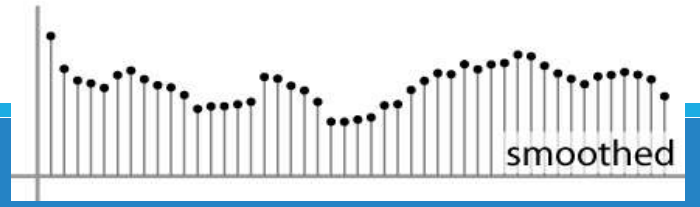
**FIGURE 3.32** Two $3 \times 3$ smoothing (averaging) filter masks. The constant multiplier in front of each mask is equal to 1 divided by the sum of the values of its coefficients, as is required to compute an average.

- One of the simplest spatial filtering operations we can perform is a smoothing operation (remove sharp features)
  - Simply average all of the pixels in a neighborhood around a central value
  - Especially useful in removing noise from images
  - Also useful for highlighting gross detail
  - Moving average in 1D:

$\frac{1}{25} \times$

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



# Average filtering: Example

$h[\cdot,\cdot]$ $\frac{1}{9}$

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

$f[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$g[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | 0 | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

# Average filtering: Example

$h[\cdot,\cdot]$ $\frac{1}{9}$

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

$f[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$g[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | 0 | 10 | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

# Average filtering: Example

$h[\cdot\,,\cdot\,]$ $\frac{1}{9}$

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

$f[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$g[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | 0 | 10 | 20 | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

# Average filtering: Example

$h[\cdot,\cdot]$ $\frac{1}{9}$

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

$f[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$g[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | 0 | 10 | 20 | 30 | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

# Average filtering: Example

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | 0 | 10 | 20 | 30 | 30 | 30 | 20 | 10 | |
| | 0 | 20 | 40 | 60 | 60 | 60 | 40 | 20 | |
| | 0 | 30 | 60 | 90 | 90 | 90 | 60 | 30 | |
| | 0 | 30 | 50 | 80 | 80 | 90 | 60 | 30 | |
| | 0 | 30 | 50 | 80 | 80 | 90 | 60 | 30 | |
| | 0 | 20 | 30 | 50 | 50 | 60 | 40 | 20 | |
| | 10 | 20 | 30 | 30 | 30 | 30 | 20 | 10 | |
| | 10 | 10 | 10 | 0 | 0 | 0 | 0 | 0 | |
| | | | | | | | | | |

# Examples: Average Filtering

(a) Original image

(b) Average filtering

(c) Using a $9 \times 9$ filter

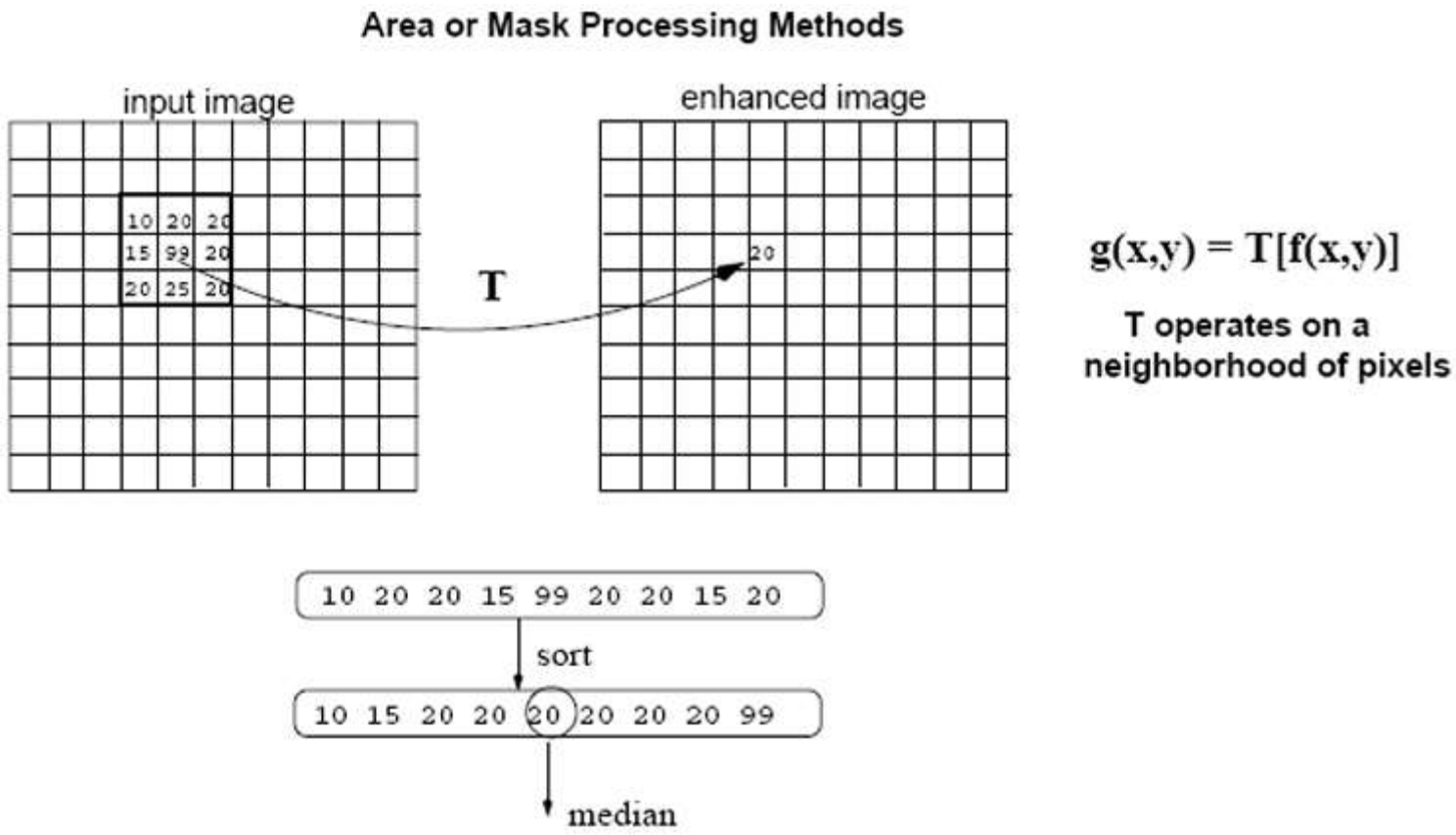
(d) Using a $25 \times 25$ filter

Original image

Smoothed image

Very Smoothed image

# Median Filtering



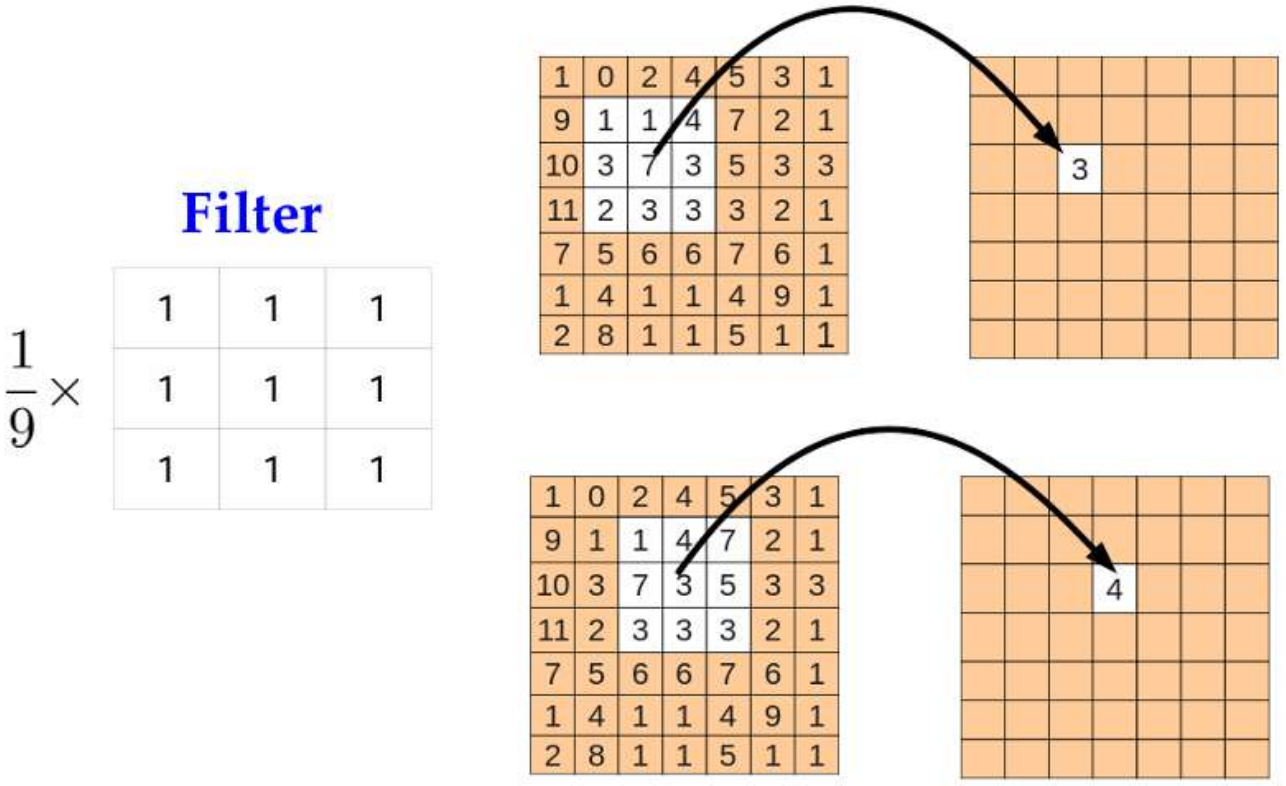
# Spatial Filtering Methods

# Spatial Filtering Methods

Main linear spatial filtering methods:

- Correlation
- Convolution

# Spatial Filtering as Correlation



# Thank you