CS370 Computer Imaging

Image Representation and Operations Part-2

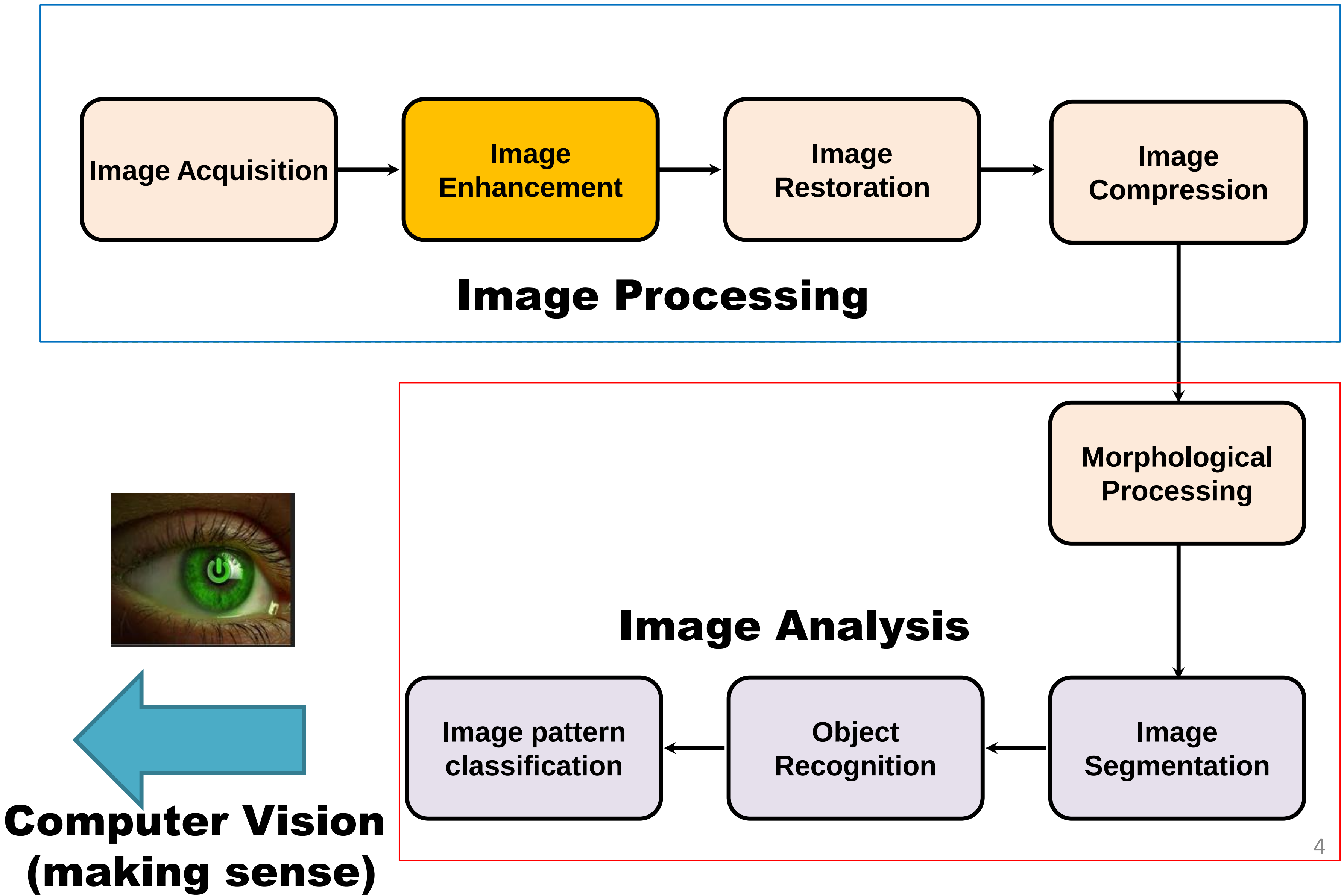
Recap

- Pixel Neighborhood
- Adjacency
- Digital Path
- Connectivity
- Region and Boundary
- Proximity Relationship
- Defining Linear Operations

Lecture Objectives

- Elementwise Versus Matrix Operations
- Operations on Images
  - Arithmetic Operations
  - Set and Logical Operations
- Spatial Operations
  - Single-pixel Operations
  - Neighborhood Operations
  - Geometric Transformations

Key Stages in DIP



Elementwise Versus Matrix Operations

Elementwise Versus Matrix Operations

- An *elementwise operation* involving one or more images is carried out on a ***pixel-by-pixel*** basis.

- For example, consider the following 2 × 2 images (matrices):

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \text{ and } \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

  – The ***elementwise product*** (often denoted using the symbol $\odot$ or $\otimes$) of these two images is:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \odot \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{bmatrix}$$

Elementwise Versus Matrix Operations

- On the other hand, the *matrix product* of the images is formed using the rules of **matrix multiplication**:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

Operations on Images

Arithmetic Operations

Arithmetic Operations

- Arithmetic operations between two images $f(x, y)$ and $g(x, y)$ are denoted as:

$$s(x,y)=f(x,y)+g(x,y)$$
$$d(x,y)=f(x,y)-g(x,y)$$
$$p(x,y)=f(x,y)\times g(x,y)$$
$$v(x,y)=f(x,y)/g(x,y)$$

- These are ***elementwise operations*** which means that they are performed between corresponding pixel pairs in f and g for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$.

- Note that image arithmetic involves images of the ***same size***.

Image Addition

- It is the pixel-wise addition of intensity values defined as:

  s(x,y)=f(x,y)+g(x,y)

- When pixel values **exceed** the **upper limit**, the values are **clipped** to a defined **maximum**:
  - [0-1] for binary images e.g. 2 becomes 1
  - [0-255] for 8 bit images e.g. 510 becomes 255

- Alternatively **rescale** the pixel values within a required range between **[a, b]** using :

$$x' = a + \frac{(x - \min(x))(b - a)}{\max(x) - \min(x)}$$

max value is **510** for 8-bit images

min value is **-255** for **8** bit images

- **Application:** noise removal by using addition and averaging.

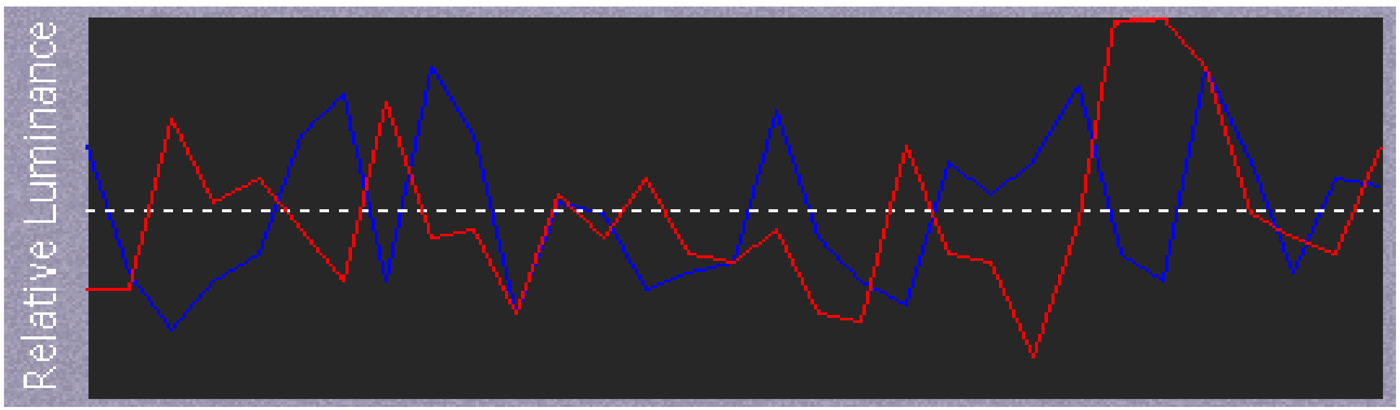
Application: Noise Removal

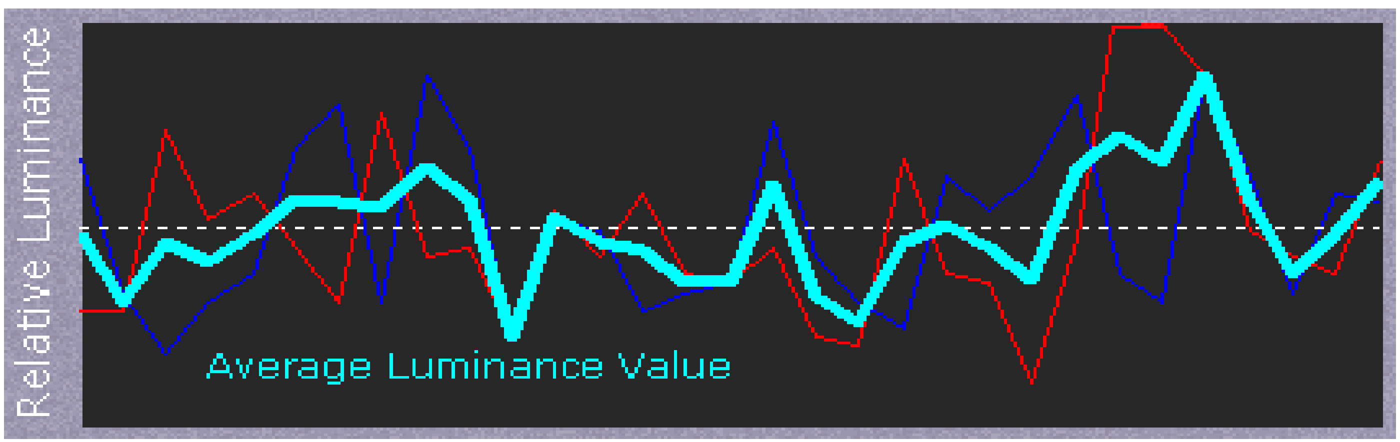
- The objective is to reduce the noise content of the output image by adding and averaging a set of noisy input images, $\{g_i(x,y)\}$ of the same scene.

- This is a technique used frequently for **image enhancement**.

Why does Adding & Averaging work?

- Image averaging works on the assumption that the noise in your image is **truly random**. This way, random fluctuations above and below actual image data will **gradually even** out as one averages more and more images.





Why does Adding & Averaging work?

- Noise is inherent in any digital sensor, but the location is not fixed !!

- If we average **enough** times, we can cancel the noise effects at a **particular location.**

Application: Noise Removal

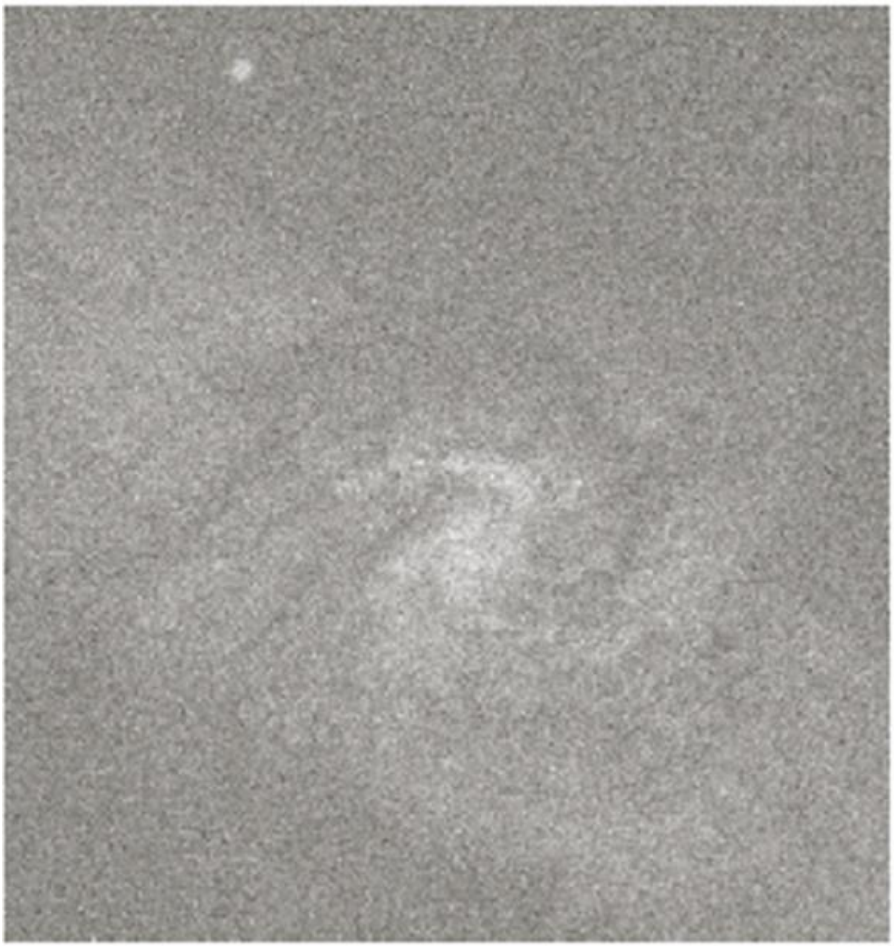


Image of Galaxy Pair NGC 3314 corrupted by additive Gaussian noise

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

where,

$\boldsymbol{\eta(x,y)}$ is the **Gaussian noise** with zero mean and a standard deviation $\boldsymbol{\sigma}$:

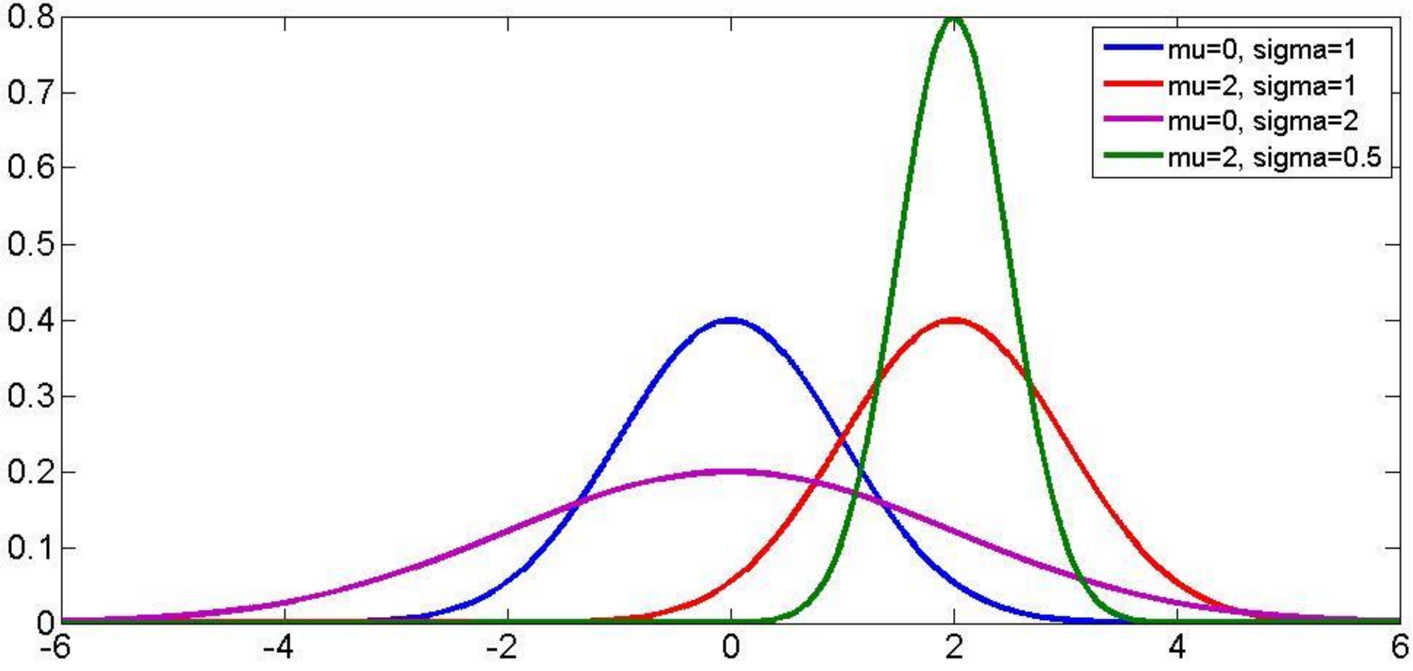
$$\eta(x,y) \sim \mathcal{N}(\mu_{\eta(x,y)}, \sigma^2_{\eta(x,y)})$$

The noise is usually introduced in **low light conditions**.

Application: Noise Removal

- **Gaussian (Normal) distribution:**

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



Application: Noise Removal

- Let's add a set of noisy images $\{g_i(x,y)\}$ and then take the average:

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

$g_i(x,y)$ is the image of the same scene

Then it follows that:

$$E\{\bar{g}(x,y)\} = f(x,y)$$

and

$$\sigma^2_{\bar{g}(x,y)} = \frac{1}{K}\sigma^2_{\eta(x,y)}$$

where $E\{\bar{g}(x,y)\}$ is the expected value of $\bar{g}(x,y)$ and $\sigma^2_{\bar{g}(x,y)}$ and $\sigma^2_{\eta(x,y)}$ are the variances of $\bar{g}(x,y)$ and $\eta(x,y)$ respectively, all at coordinates (*x*, *y*).

Application: Noise Removal

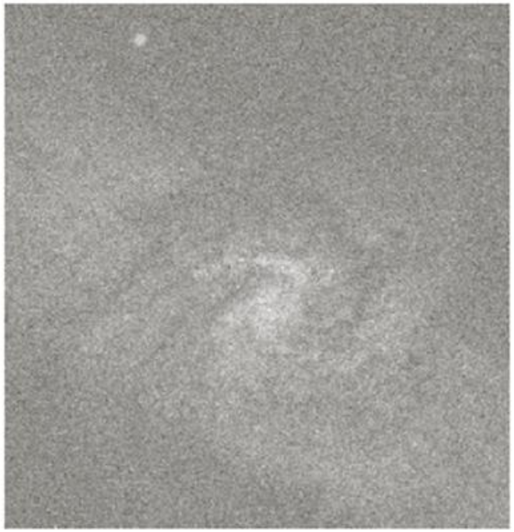
- The standard deviation (square root of the variance) at any point (*x*, *y*) in the average image is:

$$\sigma_{\bar{g}(x,y)} = \frac{1}{\sqrt{K}} \sigma_{\eta(x,y)}$$

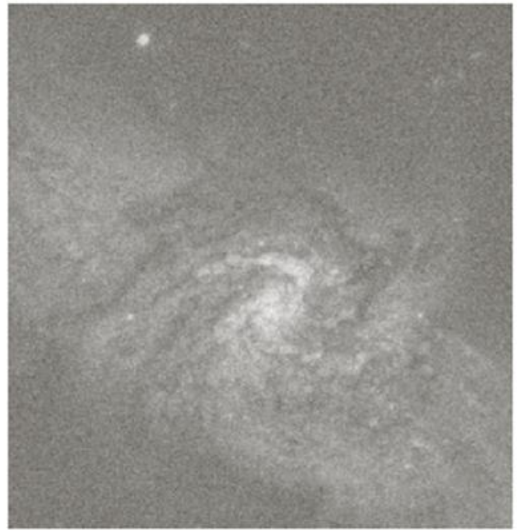
- As ***K*** increases, the variability (as measured by the variance or the standard deviation) of the pixel values at each location (*x*, *y*) decreases.

- Because $E\{\bar{g}(x,y)\} = f(x,y)$, this means that $\bar{g}(x,y)$ approaches the noiseless image *f* (*x*, *y*) as the number of noisy images used in the averaging process increases.

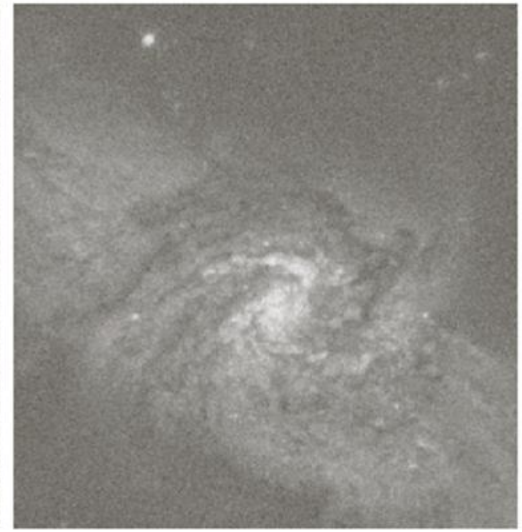
Application: Noise Removal



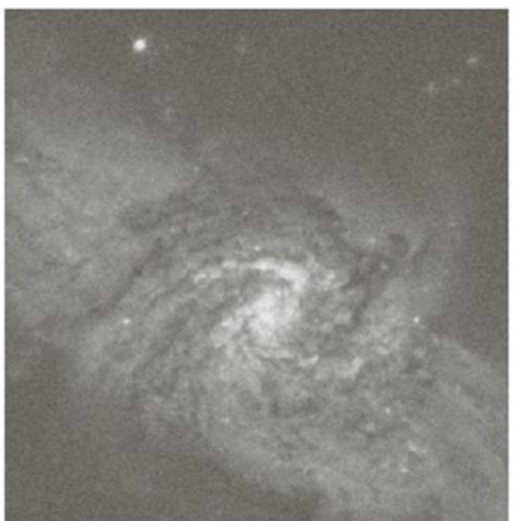
Original



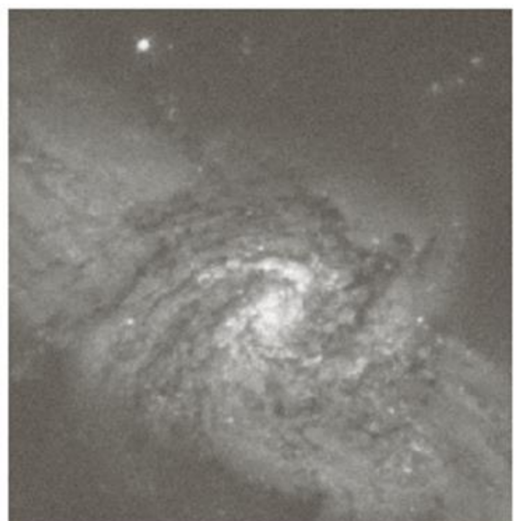
K=5



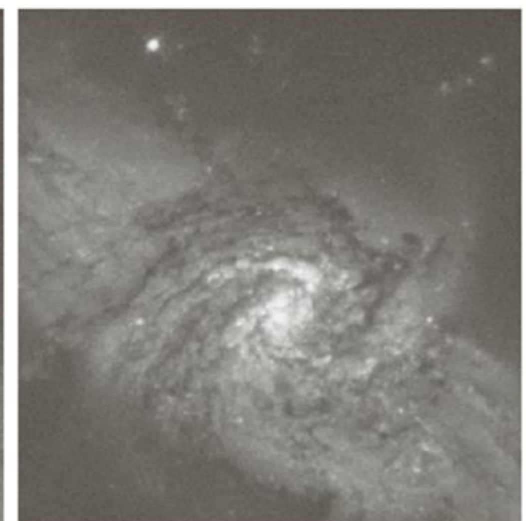
K=10



K=20



K=50



K=100

Image Subtraction

- It is the pixel-wise subtraction of intensity values defined as:

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

- When pixel values **deceed** the **lower limit**, the values are **clipped** to a defined **minimum**:
  - [0-1] for binary images e.g. -1 becomes 0
  - [0-255] for 8 bit images e.g. -255 becomes 0

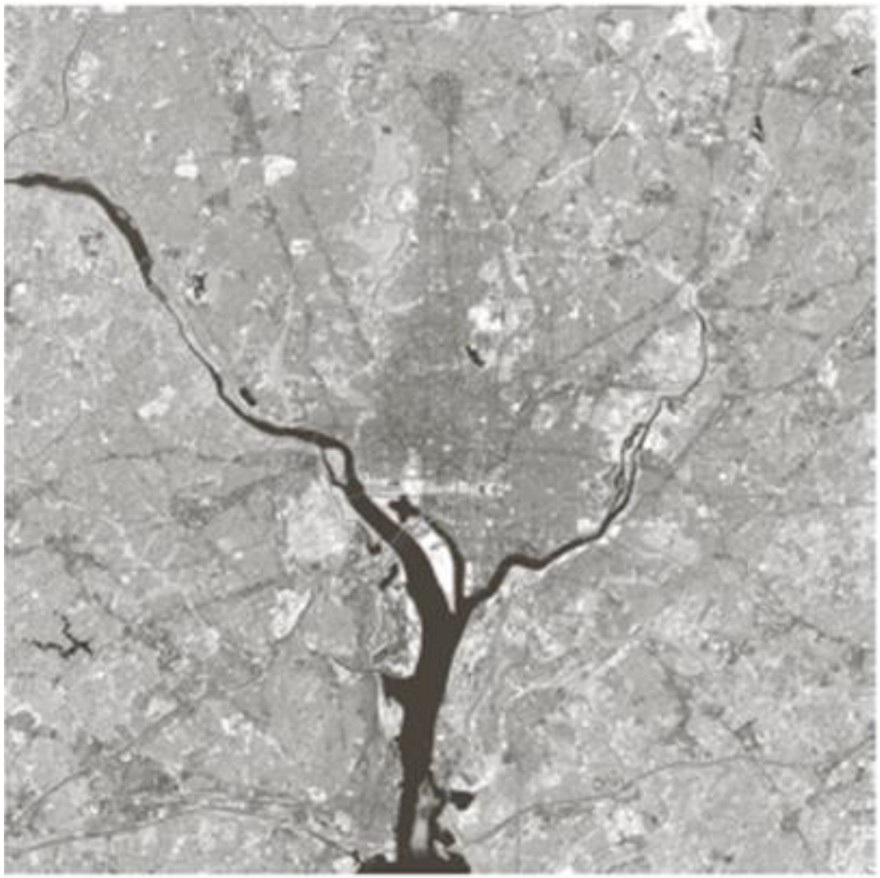
- Alternatively **rescale** the pixel values within a required range between **[a, b]** using :

$$x' = a + \frac{(x - \min(x))(b - a)}{\max(x) - \min(x)}$$

max value is **510** for 8-bit images | **min** value is **-255** for **8** bit images

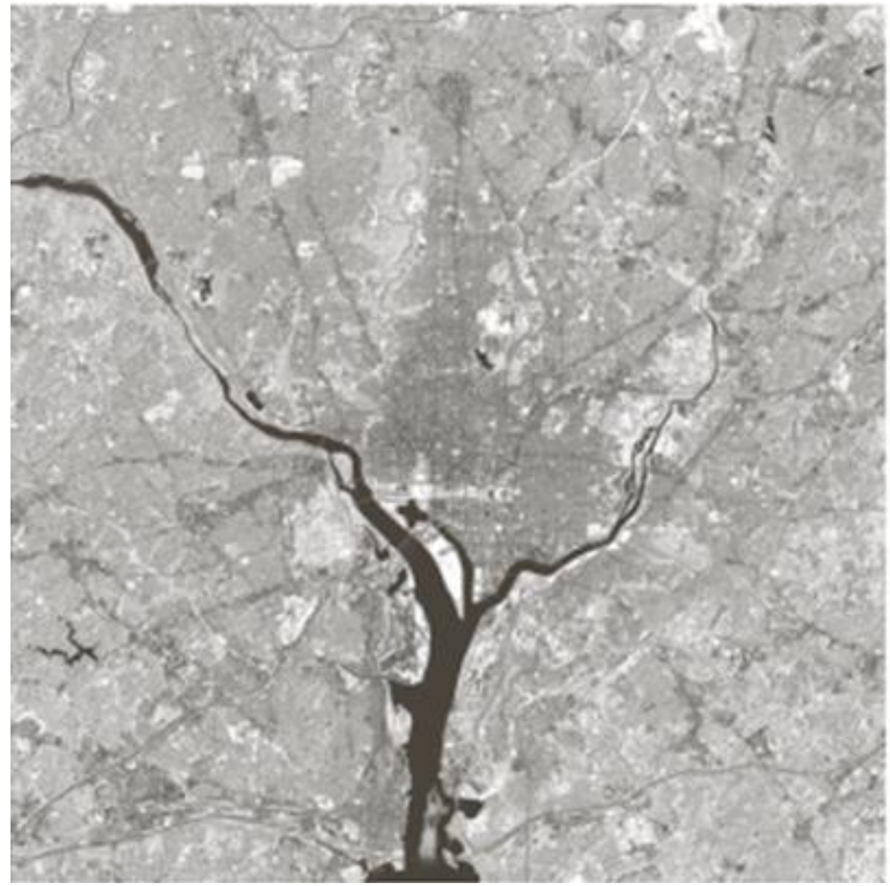
- **Application:** enhancing differences between images, extraction of details from an image.

Application: Enhancement of differences between images



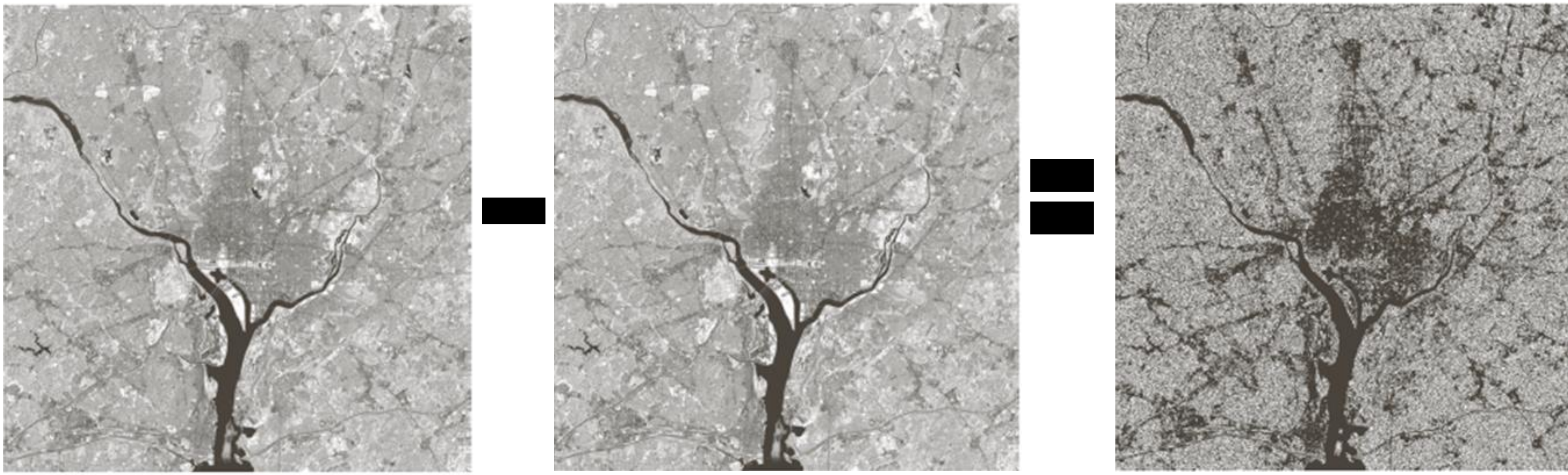
Original

Infrared image of the Washington, D.C. area



LSB is set to zero

Application: Enhancement of differences between images



Original | LSB is set to 0 | Rescaled to [0,255]

Black (0) values in the difference image indicate locations where there is no differences between the images.

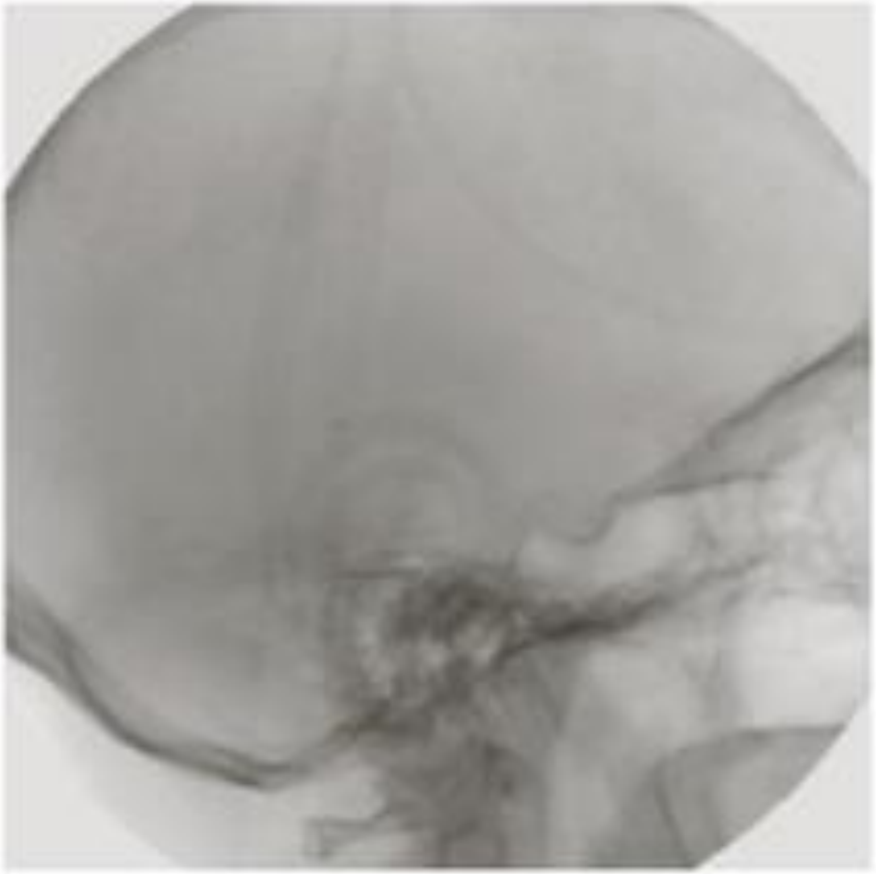
How to rescale?

Application: Extraction of details from an image (Mask Mode Radiography)

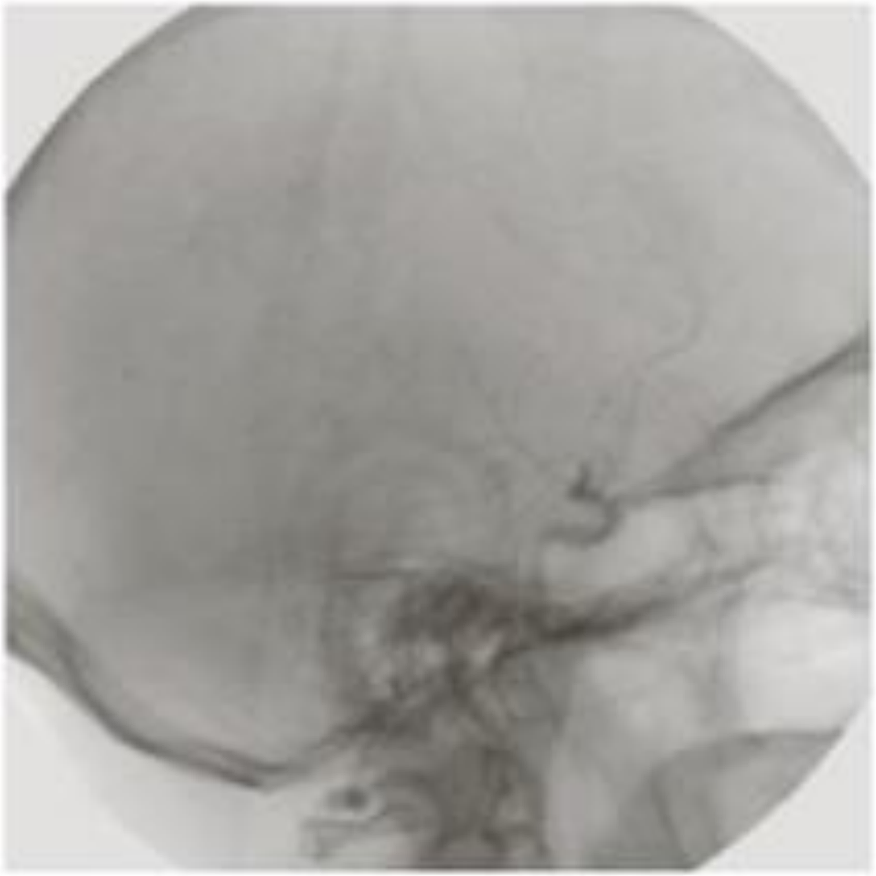
$$g(x,y)=f(x,y)-h(x,y)$$

where, $\mathbf{h(x,y)}$ is the *mask*, $\mathbf{f(x,y)}$ is the *original image* and $\mathbf{g(x,y)}$ is the *desired details*.

Application: Extraction of details from an image (Mask Mode Radiography)



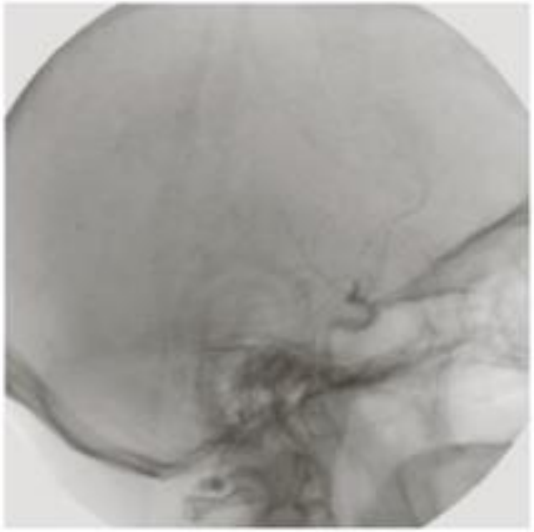
h(x,y): Mask (top of a patient's head)



f(x,y): Live image(after injecting iodine medium)

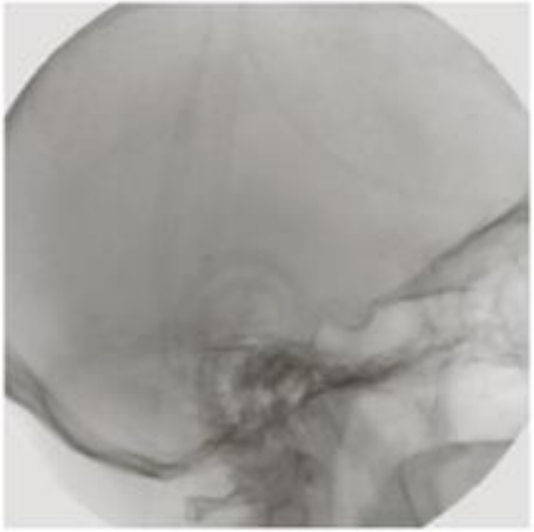
mask, is an X-ray image of a region of a patient's body captured by an intensified TV camera (instead of traditional X-ray film) located opposite an X-ray source.

Application: Extraction of details from an image (Mask Mode Radiography)



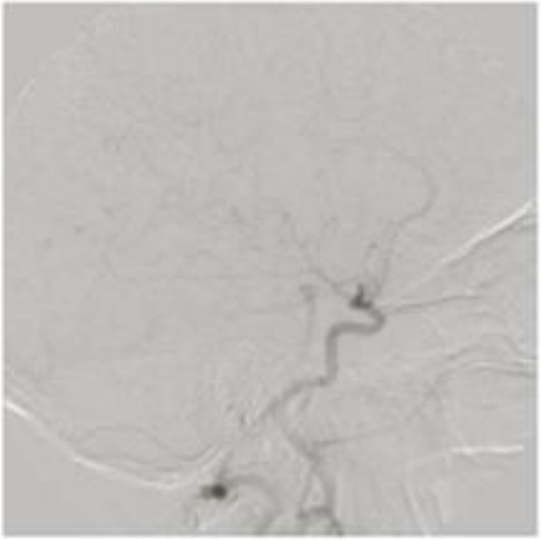
f(x,y): Live image

−



h(x,y): Mask

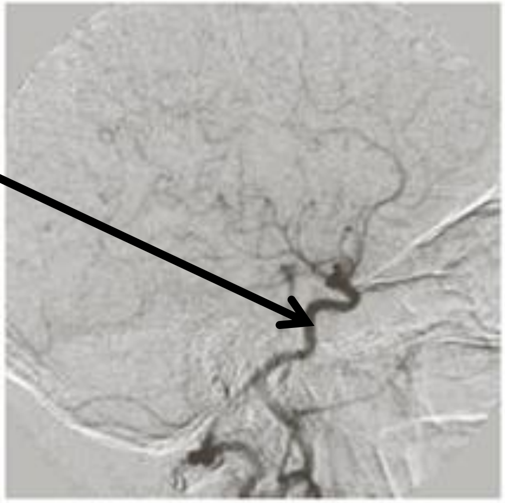
=



g(x,y): Desired details

Contrast Enhancement

Contrast Medium



“snapshot” of how contrast medium iodine is propagating through the blood vessels in the subject’s brain

Image Multiplication

- It is the pixel-wise multiplication of intensity values defined as:

  $$s(x,y)=f(x,y)\times g(x,y)$$

- When pixel values **exceed** the **upper limit**, the values are **clipped** to a defined **maximum:**
  - [0-1] for binary images e.g. 2 becomes 1
  - [0-255] for 8 bit images e,g. 510 becomes 255

- Alternatively **rescale** the pixel values within a required range between **[a, b]** using :

  $$x' = a + \frac{(x - \min(x))(b-a)}{\max(x) - \min(x)}$$

  max value is **510** for 8-bit images

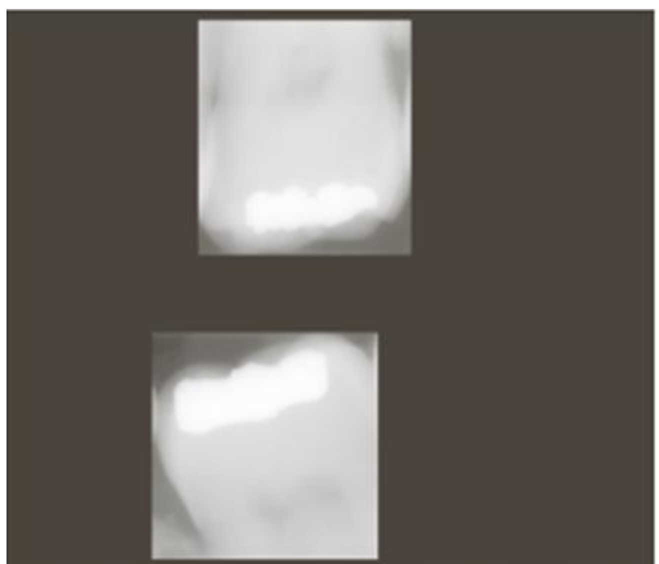
  min value is **-255** for **8** bit images

- **Application:** masking, shading correction.

Application: Masking

Dental X-ray

Region of interest (ROI) mask



Product

Image Division

- It is the pixel-wise division of intensity values defined as:

$$s(x,y)=f(x,y) \div g(x,y)$$

- When performing division, we have the extra requirement that a small number should be **added** to the pixels of the divisor image to avoid **division by 0 error**.

- **Application:** shading correction.

Application: Shading Correction

- $g(x,y)=f(x,y)\times h(x,y)$, where **f(x,y)** is the *perfect image*, **h(x,y)** is known *shading function* and **g(x,y)** is the *result* of image acquisition.

f(x,y): Original image

h(x,y): Shading function

g(x,y): Shaded image

f(x,y) is a microscopy image of a tungsten filament and support, magnified 130 times

***h*(*x*, *y*)** is assumed to be *known* or can be *estimated*

Application: Shading Correction

$g(x,y)$: **Shaded image**

÷

$h(x,y)$: **Shading pattern**

=

$f(x,y)$: **Original image**

***h*(*x*, *y*)** is assumed to be *known* or can be *estimated*

Capturing Full Range of Values in an Image

- Given a digital image ***g*** resulting from one or more arithmetic (or other) operations, an approach guaranteeing that the *full range of a values is "captured"* into a fixed number of bits is as follows:

Step-1: Transform into an image whose *minimum value is zero*.

$$g_m = g - \min(g)$$

Step-2: Scale the transformed image in the range [0, K].

$$g_s = K\left[g_m / \max(g_m)\right]$$

15	**14**	**236**
21	**1**	**22**
22	**32**	**3**

- **g** is the original image and $\mathbf{g_s}$ is the scaled image
- K =255 for 8-bit image

Scale range [0, 200]

Scaling Pixel Values

Rescale the pixel values within a required range between **[a, b]**

$$x' = a + \frac{(x - \min(x))(b - a)}{\max(x) - \min(x)}$$

15	**14**	**236**
21	**1**	**22**
22	**32**	**3**

Scale range [0, 200]

Capturing **full range of values** in an Image

$$g_m = g - \min(g)$$

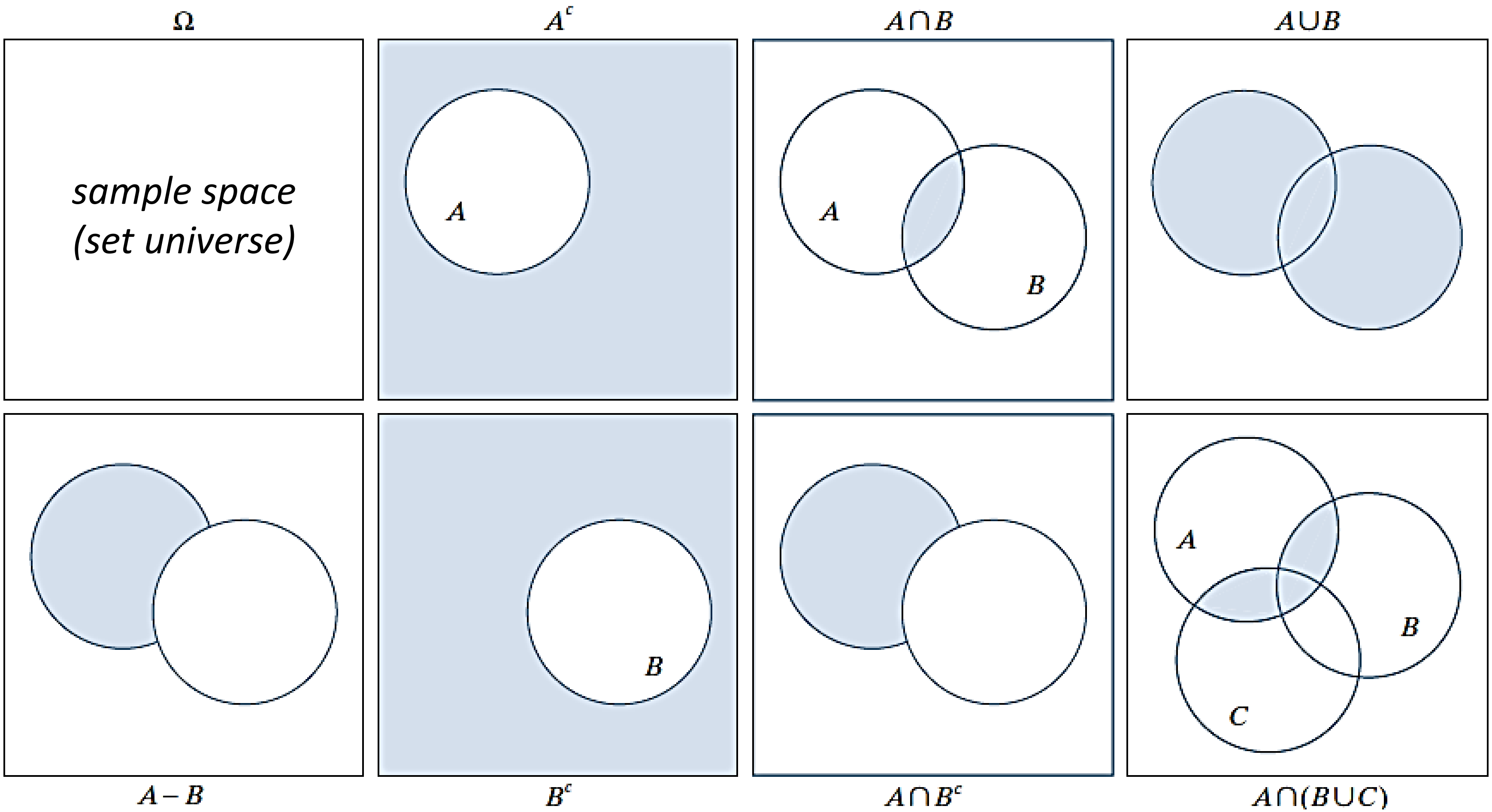
$$g_s = K\left[g_m / \max(g_m)\right]$$

15	**14**	**236**
21	**1**	**22**
22	**32**	**3**

Scale range [0, 200]

Set and Logical Operations

Venn diagrams



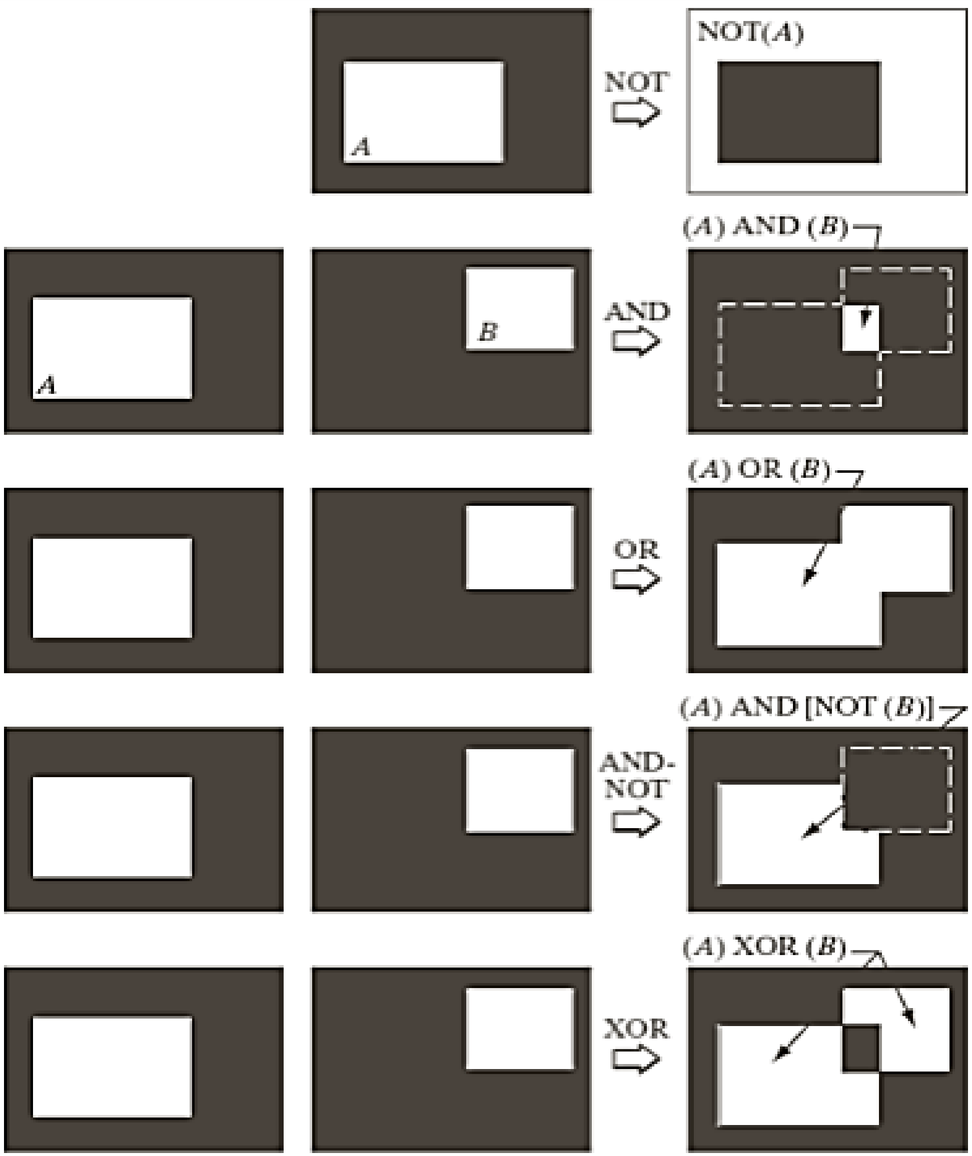
Binary Image – set operations

- Let $A=\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$ and $B=\{(x'_1,y'_1),(x'_2,y'_2),\ldots,(x'_n,y'_n)\}$ be two **binary** images where **x** and **y** are *spatial coordinates* .

Then,

- **A∪B** $= \{(x,y) \mid (x,y)\in A$ **OR** $(x,y)\in B\}$
- **A∩B** $= \{(x,y) \mid (x,y)\in A$ **AND** $(x,y)\in B\}$
- **A^c** $= \{(x,y) \mid (x,y)\notin A\}$
- **A-B** $= \{(x,y) \mid (x,y)\in A$ **AND** $(x,y)\notin B\}$

Binary Image – set operations



Foreground = white (1)

Background = black (0)

AND = (A∩B)

OR = (A∪B)

NOT = (A) ^{c}

XOR = (A∩B^{c})∪(A^{c}∩B)

Grayscale Image – set operations

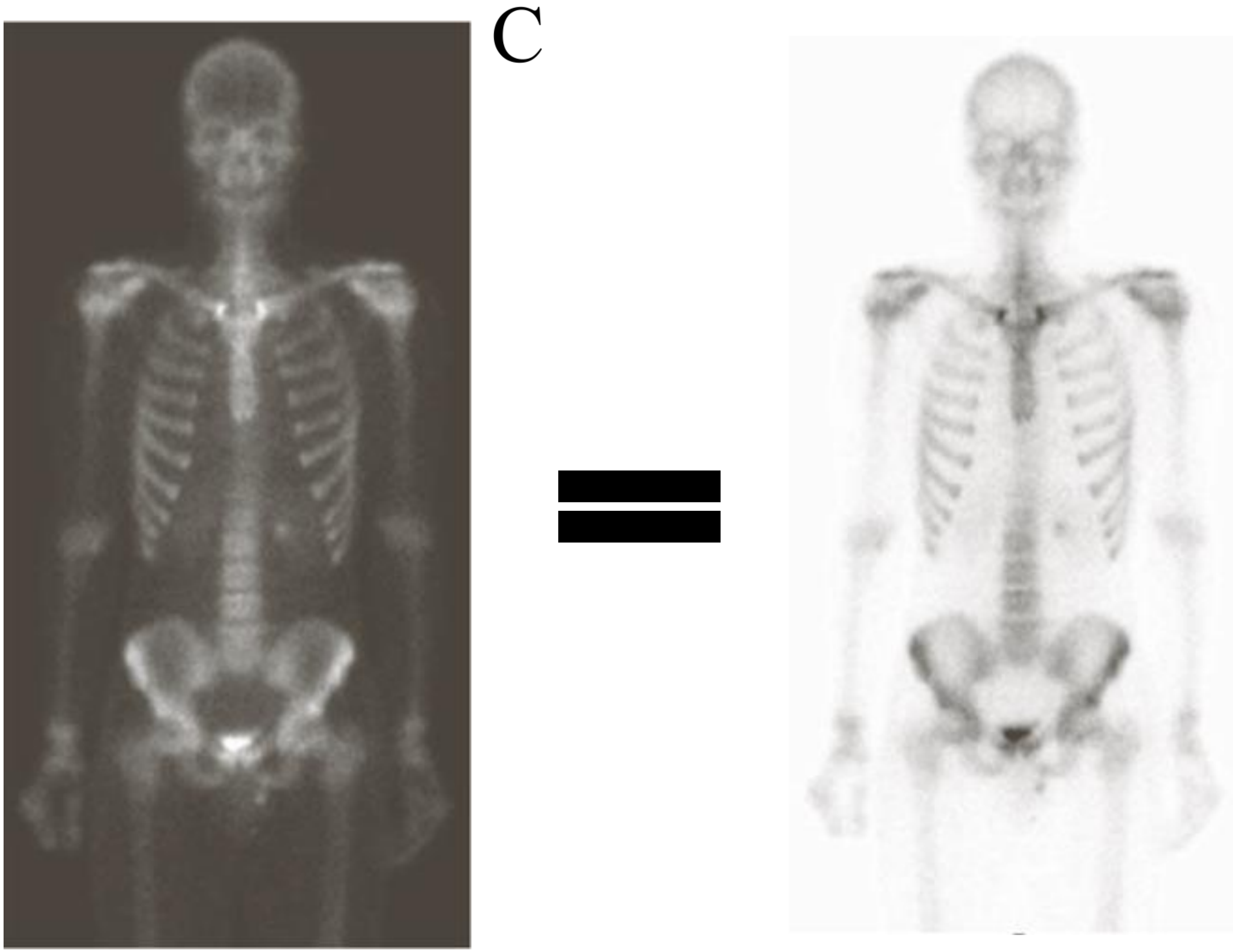
- Let $A=\{(x_1,y_1,z_1)(x_2,y_2,z_2),\ldots,(x_n,y_n,z_n)\}$ and $B=\{(x'_1,y'_1,z'_1),(x'_2,y'_2,z'_2),\ldots,(x'_n,y'_n,z'_n)\}$ be two **gray-scale** images where **x** and **y** are *spatial coordinates* and **z** denotes *intensity values* at coordinates (x,y).

Then,

- **A∪B**$=\{(x,y,z)\mid(x,y,z_1)\in A,(x,y,z_2)\in B,\ \mathbf{z=max(z_1,z_2)}\}$
- **A∩B**$=\{(x,y,z)\mid(x,y,z_1)\in A,(x,y,z_2)\in B,\ \mathbf{z=min(z_1,z_2)}\}$
- **A-B**$=\{(x,y,z)\mid(x,y,z_1)\in A,(x,y,z_2)\in B,\ \mathbf{z=(z_1-z_2)}\}$
- $\mathbf{A^c}=\{(x,y,K-z)\mid(x,y,z)\in A\}$

  where ***K*** is a constant equal to the maximum intensity value 2^k-1 in the image, where k is the number of bits used to represent *z*. For 8-bit image, K =255.

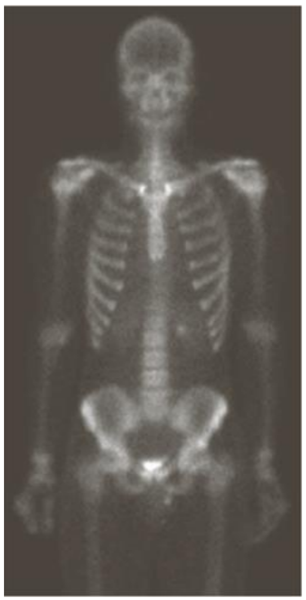
Gray-scale Image (obtaining image negative)



f(x,y)

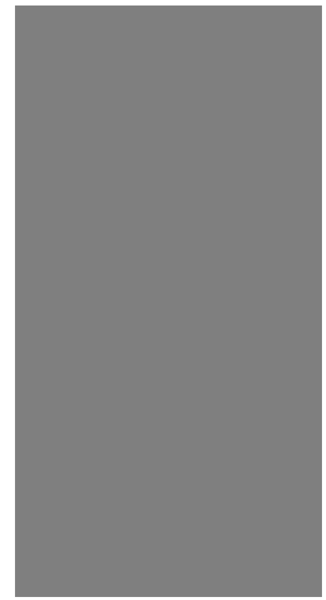
$A^c = \{(x, y, 255 - z) \mid (x, y, z) \in A\}$

Gray-scale Image (highlighting image parts)



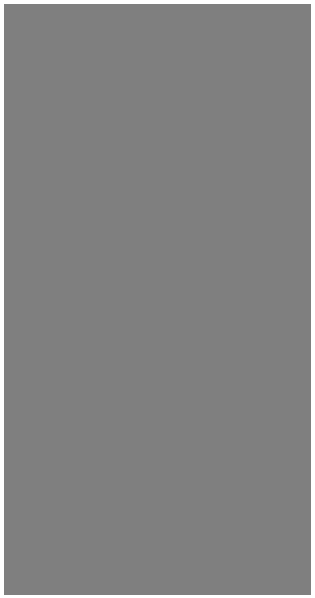
f(x,y)

$$3 \times z_{mean} f(x,y) =$$



g(x,y) = 3 times the mean intensity of f(x,y)

Gray-scale Image (highlighting image parts)



$\cup$

f(x,y)

$=$

Highlighted image parts

g(x,y) = 3 times the mean intensity of f(x,y)

$$A \cup B = \{(x,y,z) \mid (x,y,z_1) \in A, (x,y,z_2) \in B, \mathbf{z = \max(z_1, z_2)}\}$$

Spatial Operations

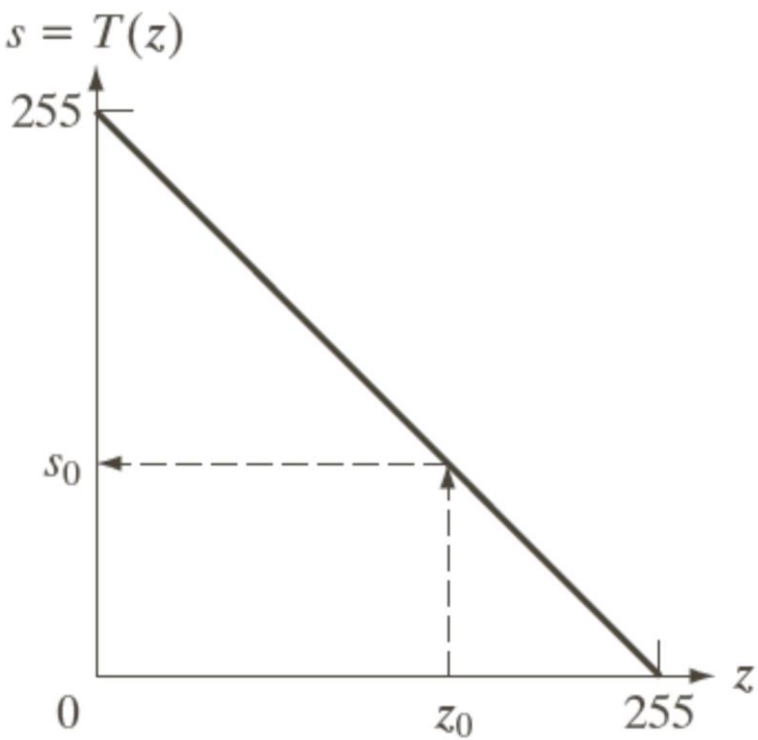
Image Spatial Operations

- **Transform:** is a *mathematical function* or *formula* or *a matrix* that takes an intensity value and returns another valid intensity value.

- Spatial operations are performed *directly on the pixels* of an image.
  1. Single-pixel operations
  2. Neighborhood operations (Spatial filtering)
  3. Geometric spatial transformations

Single-pixel Operations

- Alter the intensity of pixels <u>*individually*</u> using a transformation function **`s=T(z)`**.



Transformation function

f(x,y)

image negative

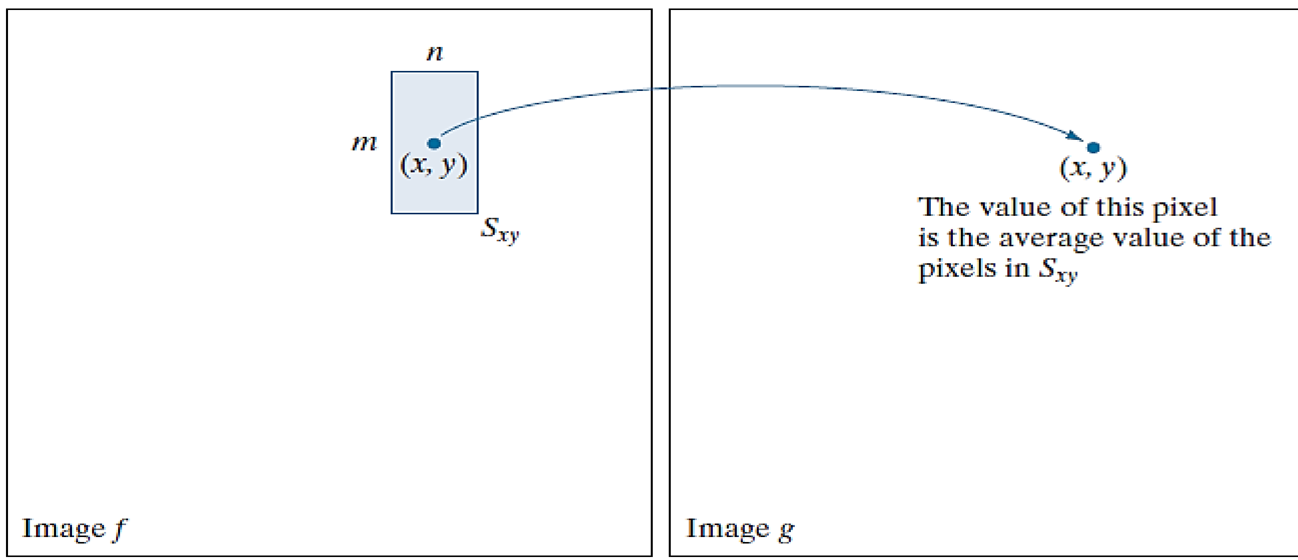
Neighborhood Operations

- Let $\boldsymbol{S_{xy}}$ denote the set of coordinates of a neighborhood centered on an arbitrary point $\boldsymbol{(x,y)}$ in an image $\boldsymbol{f}$.

- Neighborhood processing generates a corresponding pixel at the **same coordinates** in an output (processed) image, $\boldsymbol{g}$, such that the value of that pixel is determined by a <u>*specified operation on the neighborhood of pixels*</u> in the input image $\boldsymbol{f}$ with coordinates in the set $\boldsymbol{S_{xy}}$.

(x-1, y-1)	(x-1, y)	(x-1, y+1)
(x, y-1)	p(x,y)	(x, y+1)
(x+1, y-1)	(x+1, y)	(x+1, y+1)

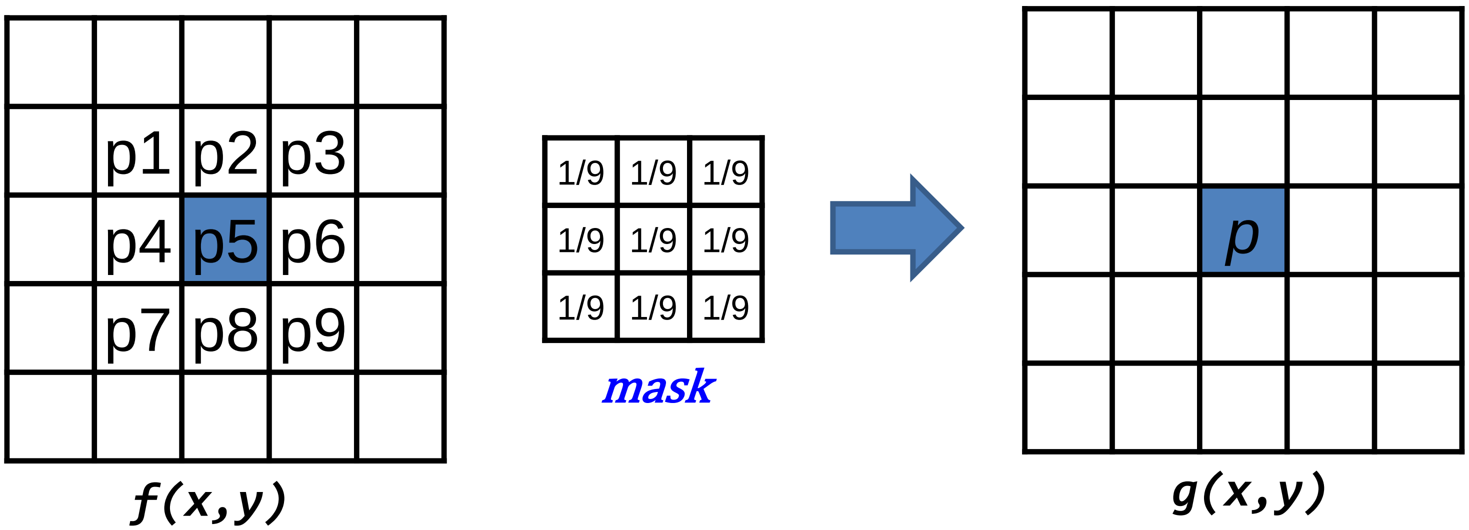
$\boldsymbol{S_{xy}}$



$$g(x,y) = \frac{1}{mn} \sum_{(r,c)\in S_{xy}} f(r,c)$$

Neighborhood Operations

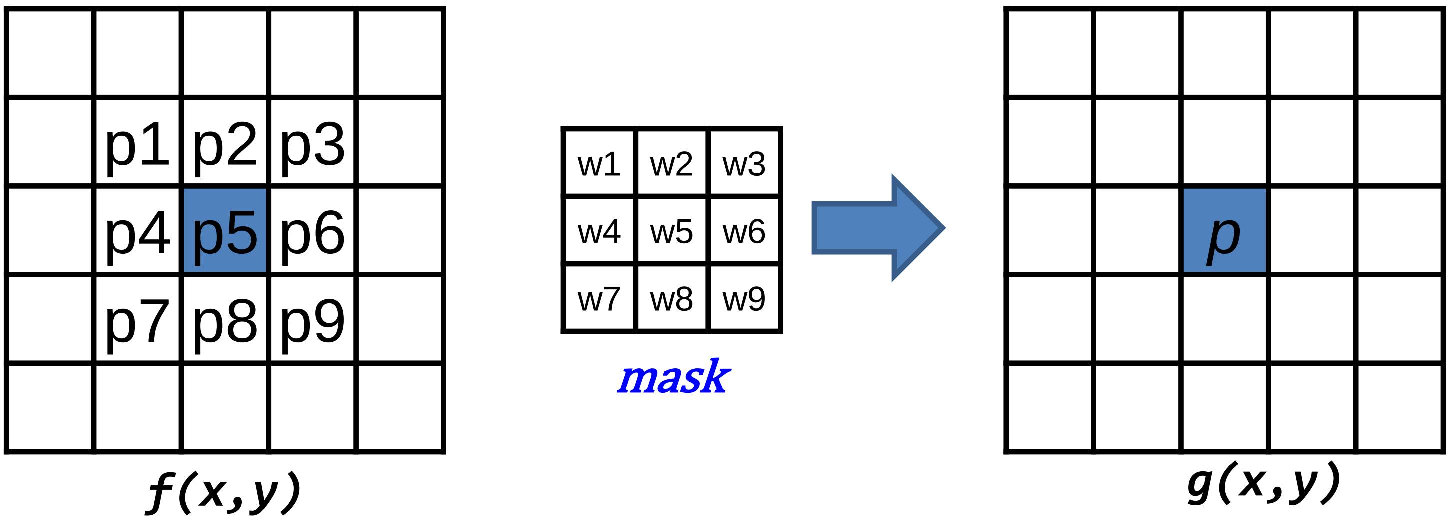
- Uses a **"mask"** (also known as "filter", "kernel", "window")



$$p = \frac{1}{9}(p_1 + p_2 + \cdots + p_9)$$

Neighborhood Operations

- Generalized Weights



$$p = \sum_{i=1}^{9} w_i p_i$$

- In the previous example: $w_i = 1/9$

Neighborhood Operations – local blurring

An aortic angiogram

$$g(x,y)=\frac{1}{mn}\sum_{(r,c)\in S_{xy}} f(r,c)$$ *for m*=*n*=41

Eliminate small details and render “blobs” corresponding to the largest regions of an image

Next Lecture

- Spatial Operations
  - Geometric spatial transformations
- Image Interpolation
- Image Registration
- Image Domain Transforms