CPE409 Image Processing

# Part 3
# Basic Concepts of Image Processing

Assist. Prof. Dr. Caner ÖZCAN

Those who wish to succeed must ask the right preliminary questions.
~Aristotle

# Outline

2. Digital Image Fundamentals
   - Elements of Visual Perception
   - Light and the Electromagnetic Spectrum
   - Image Sensing and Acquisition
   - Image Sampling and Quantization
   - Some Basic Relationships Between Pixels
   - An Introduction to the Mathematical Tools Used in Digital Image Processing
   - MATLAB Image Processing

# Some Basic Relationships Between Pixels

- **Neighbors** of a pixel p at coordinates (x,y)

- **4-neighbors of p**, denoted by $\mathbf{N_4}$ **(p):**
  (x-1, y), (x+1, y), (x,y-1), and (x, y+1).

- **4 diagonal neighbors of p,** denoted by $\mathbf{N_D}$ **(p):**
  (x-1, y-1), (x+1, y+1), (x+1,y-1), and (x-1, y+1).

- **8 neighbors of p,** denoted $\mathbf{N_8}$**(p)**
  $N_8(p) = N_4(p) \cup N_D(p)$

# Some Basic Relationships Between Pixels

- **Adjacency**

  Let V be the set of intensity values.

  In a binary image, if we talk of the adjaceny of the 1-valued pixels, V = {1}.

- **4-adjacency:** Two pixels p and q with values from V are 4-adjacent if q is in the set $N_4(p)$.

- **8-adjacency:** Two pixels p and q with values from V are 8-adjacent if q is in the set $N_8(p)$.

# Some Basic Relationships Between Pixels

- **Adjacency**

  Let V be the set of intensity values

  At the adjacency of the gray level pixels, the V set is any subset of this 256 value.

- **m-adjacency:** Two pixels p and q with values from V are m-adjacent if

  (i) q is in the set $N_4(p)$, or

  (ii) q is in the set $N_D(p)$ and the set $N_4(p) \cap N_4(q)$ has no pixels whose values are from V.

# Some Basic Relationships Between Pixels

## Path

- A (digital) path (or curve) from pixel p with coordinates $(x_0, y_0)$ to pixel q with coordinates $(x_n, y_n)$ is a sequence of distinct pixels with coordinates

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$$

  Where $(x_i, y_i)$ and $(x_{i-1}, y_{i-1})$ are adjacent for $1 \le i \le n$.

- Here $n$ is the *length* of the path.
- If $(x_0, y_0) = (x_n, y_n)$, the path is ***closed*** path.
- We can define 4-, 8-, and m-paths based on the type of adjacency used.

# Examples: Adjacency and Path

**V = {1}**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | | 0 | 1 | 1 | | 0 | 1 | 1 |
| 0 | 1 | 0 | | 0 | 1 | 0 | | 0 | 1 | 0 |
| 0 | 0 | 1 | | 0 | 0 | 1 | | 0 | 0 | 1 |

# Examples: Adjacency and Path

**V = {1}**

$$\begin{matrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \qquad \begin{matrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \qquad \begin{matrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

**8-adjacent**

# Examples: Adjacency and Path

**V = {1}**

$$\begin{matrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \qquad \begin{matrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \qquad \begin{matrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

**8-adjacent** **m-adjacent**

# Examples: Adjacency and Path

**V = {1, 2}**

$$\begin{array}{ccc} 0_{1,1} & 1_{1,2} & 1_{1,3} \\ 0_{2,1} & 1_{2,2} & 0_{2,3} \\ 0_{3,1} & 0_{3,2} & 1_{3,3} \end{array}$$

$$\begin{array}{ccc} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}$$

**8-adjacent**

$$\begin{array}{ccc} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}$$

**m-adjacent**

The 8-path from (1,3) to (3,3):

(i) (1,3), (1,2), (2,2), (3,3)

(ii) (1,3), (2,2), (3,3)

The m-path from (1,3) to (3,3):

(1,3), (1,2), (2,2), (3,3)

# Distance Measures

- Given pixels *p, q* and *z* with coordinates (x, y), (s, t), (u, v) respectively, the distance function D has following properties:

  - $D(p, q) \geq 0$ [$D(p, q) = 0$, iff $p = q$]
  - $D(p, q) = D(q, p)$
  - $D(p, z) \leq D(p, q) + D(q, z)$

# Distance Measures

- The following are the different Distance measures:

a. Euclidean Distance :

$D_e(p, q) = [(x-s)^2 + (y-t)^2]^{1/2}$

b. City Block Distance:

$D_4(p, q) = |x-s| + |y-t|$

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

c. Chess Board Distance:

$D_8(p, q) = \max(|x-s|, |y-t|)$

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

# Question 5

- **In the following arrangement of pixels, what's the value of the chessboard distance between the circled two points?**

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

# Question 6

- **In the following arrangement of pixels, what's the value of the city-block distance between the circled two points?**

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

# Introduction to Mathematical Operations Used in Digital Image Processing

## ▶ Array vs. Matrix Operation

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \qquad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

Array product operator

$$A .* B = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{bmatrix}$$

**Array product**

Matrix product operator

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

**Matrix product**

# Introduction to Mathematical Operations Used in Digital Image Processing

- **Linear and Non-Linear Operations**

$$H\left[f(x,y)\right]=g(x,y)$$

- If

$$\begin{aligned}
&H\left[a_i f_i(x,y)+a_j f_j(x,y)\right]\\
&=H\left[a_i f_i(x,y)\right]+H\left[a_j f_j(x,y)\right] && \text{Toplanabilirlik}\\
&=a_i H\left[f_i(x,y)\right]+a_j H\left[f_j(x,y)\right] && \text{Homojenlik}\\
&=a_i g_i(x,y)+a_j g_j(x,y)
\end{aligned}$$

then H is called a **linear operator**. If H has the above Attribute If not, it is expressed as a **nonlinear operator**.

# Arithmetic Operations

- Arithmetic operations between images are array operations. The four arithmetic operations are denoted as

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

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

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

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

# Example: Addition of Noisy Images for Noise Reduction

Noiseless image: f(x,y)

Noise: n(x,y) (at every pair of coordinates (x,y), the noise is uncorrelated and has zero average value)

Corrupted image: g(x,y)

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

Reducing the noise by adding a set of noisy images, $\{g_i(x,y)\}$

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

# Example: Addition of Noisy Images for Noise Reduction

- In astronomy, imaging under very low light levels frequently causes sensor noise to render single images virtually useless for analysis.

- In astronomical observations, similar sensors for noise reduction by observing the same scene over long periods of time. Image averaging is then used to reduce the noise.

# Example: Addition of Noisy Images for Noise Reduction

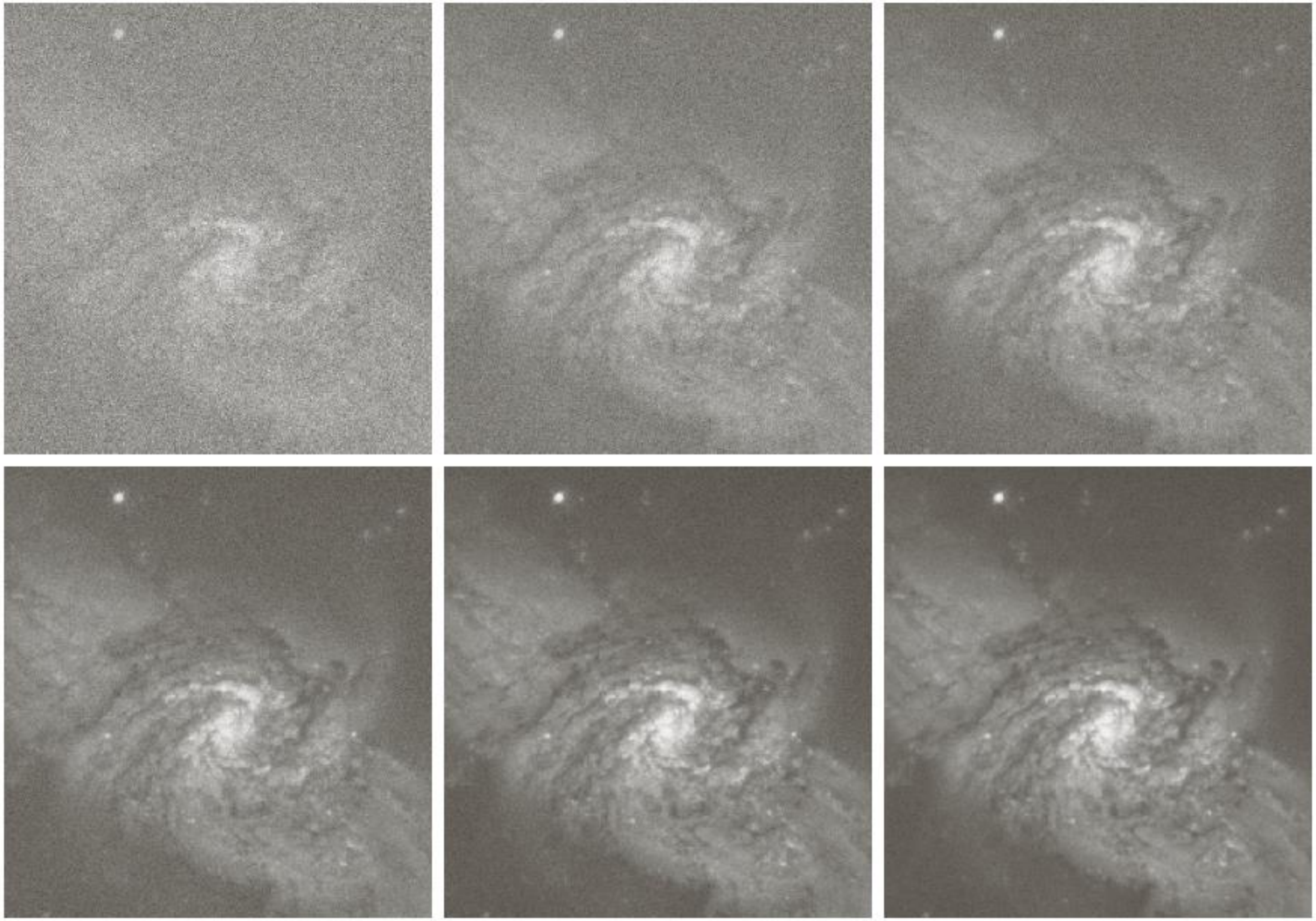



**FIGURE 2.26** (a) Image of Galaxy Pair NGC 3314 corrupted by additive Gaussian noise. (b)–(f) Results of averaging 5, 10, 20, 50, and 100 noisy images, respectively. (Original image courtesy of NASA.)

# An Example of Image Subtraction: Mask Mode Radiography

**Mask $h(x,y)$:** an X-ray image of a region of a patient's body

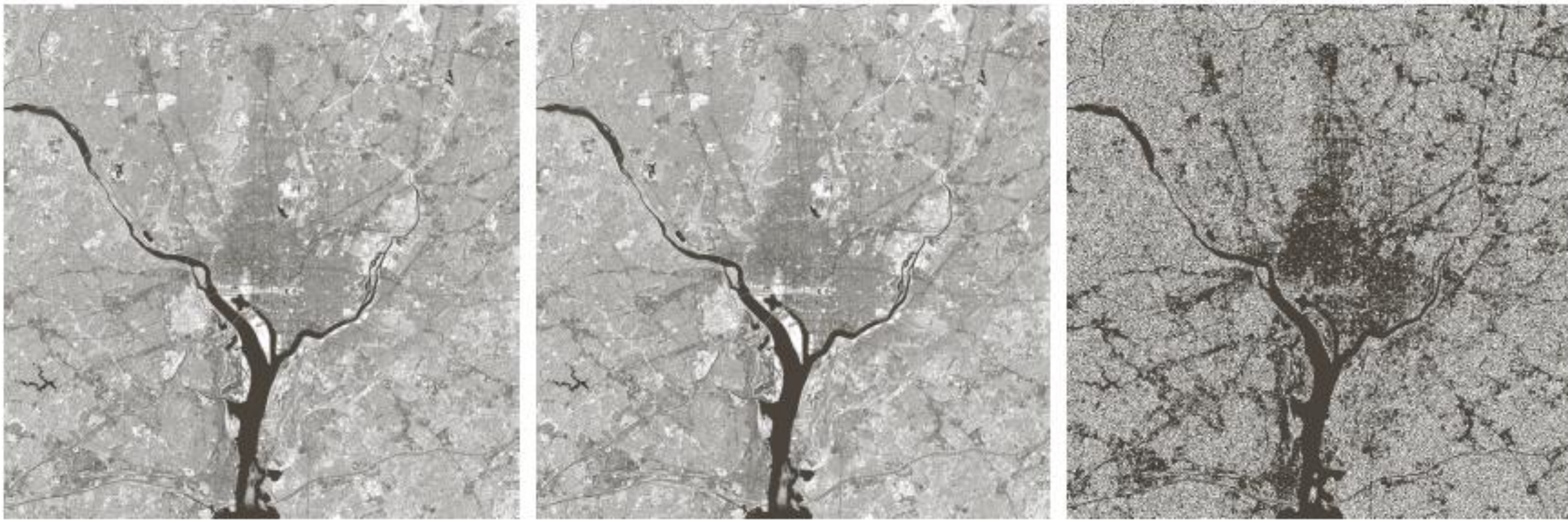
**Live images $f(x,y)$:** X-ray images captured at TV rates after injection of the contrast medium

**Enhanced detail $g(x,y)$**

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

The procedure gives a movie showing how the contrast medium propagates through the various arteries in the area being observed.

# An Example of Image Subtraction: Mask Mode Radiography



a b c

**FIGURE 2.27** (a) Infrared image of the Washington, D.C. area. (b) Image obtained by setting to zero the least significant bit of every pixel in (a). (c) Difference of the two images, scaled to the range [0, 255] for clarity.

# An Example of Image Subtraction: Mask Mode Radiography

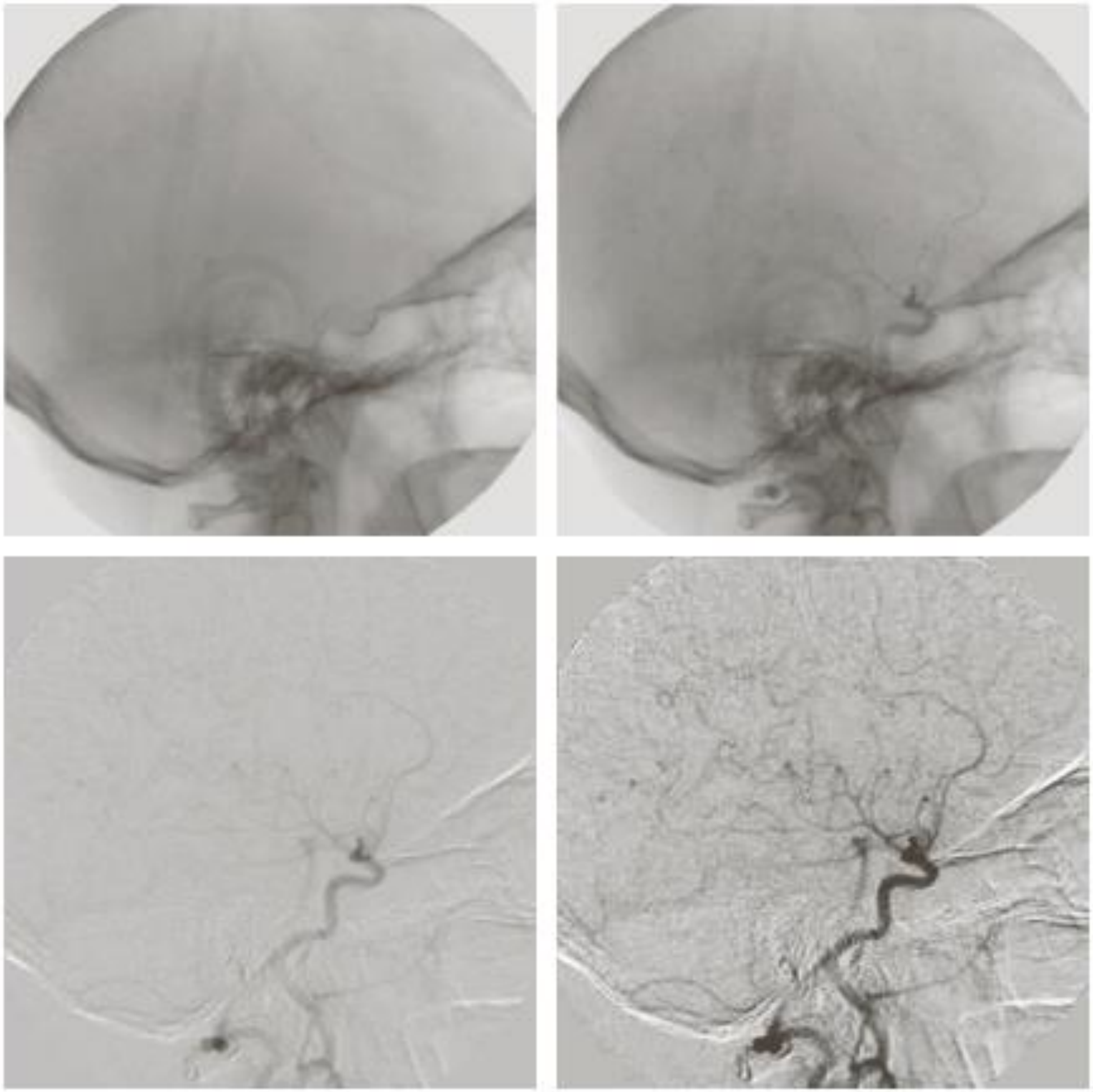



**FIGURE 2.28** Digital subtraction angiography. (a) Mask image. (b) A live image. (c) Difference between (a) and (b). (d) Enhanced difference image. (Figures (a) and (b) courtesy of The Image Sciences Institute, University Medical Center, Utrecht, The Netherlands.)

# An Example of Image Multiplication

a b c

**FIGURE 2.29** Shading correction. (a) Shaded SEM image of a tungsten filament and support, magnified approximately 130 times. (b) The shading pattern. (c) Product of (a) by the reciprocal of (b). (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

# An Example of Image Multiplication

a b c

**FIGURE 2.30** (a) Digital dental X-ray image. (b) ROI mask for isolating teeth with fillings (white corresponds to 1 and black corresponds to 0). (c) Product of (a) and (b).

# Python Image Processing

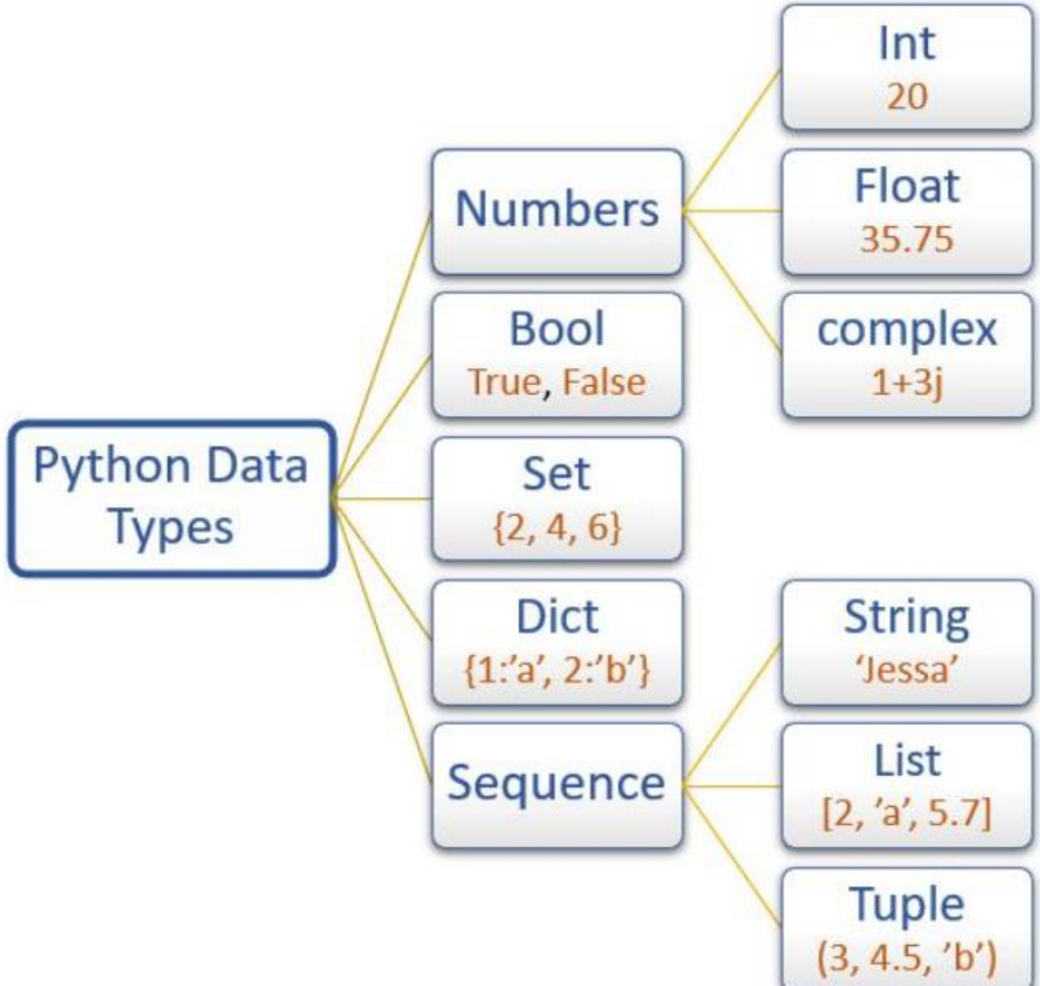
- The Image Processing Toolbox is a collection of functions that extend the capabilities of Python's digital computing environment. The toolbox supports a wide variety of image processing operations, including:
  - Geometric operations
  - Neighborhood and block transactions
  - Linear filtering and filter design
  - Transformations
  - Image analysis and enhancement
  - Binary image operations
  - Region of interest operations

# Python Image Processing

- Python can import/export many image formats:
  - BMP (Microsoft Windows Bitmap)
  - GIF (Graphics Interchange Files)
  - HDF (Hierarchical Data Format)
  - JPEG (Joint Photographic Experts Group)
  - PCX (Paintbrush)
  - PNG (Portable Network Graphics)
  - TIFF (Tagged Image File Format)
  - XWD (X Window Dump)
  - raw-data ve diğer görüntü verisi tipleri

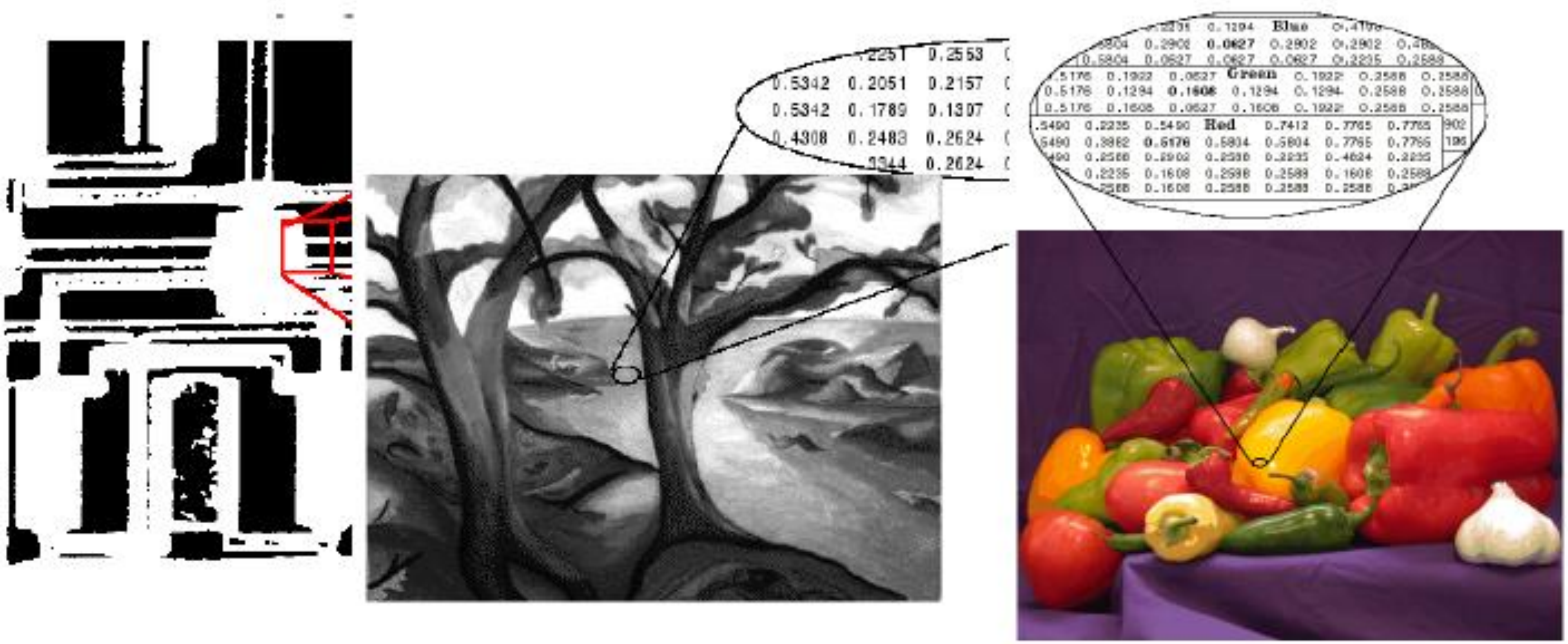
# Python Image Processing

- Python data types:



# Python Image Processing

- Binary images : {0,1}
- Intensity images : [0,1] or `uint8`, `double` etc.
- RGB images : m × n × 3
- Multidimensional images: m × n × p (p is the number of layers)



# Python Image Processing

- Reading and writing images in Python

ode + Markdown | Run All Clear Outputs of All C

```
import cv2
import numpy as np

resim1 = cv2.imread("brain.jpg")

cv2.imshow("Brain",resim1)

#Resmin boyutunu öğrenmek için:
print(resim1.size)

#Resmin veri tipini öğrenmek için:
print(resim1.dtype)

#Resmin genişliğini, yüksekliğini ve kaç kanaldan oluştuğunu öğrenmek için:
print(resim1.shape)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

17.9s

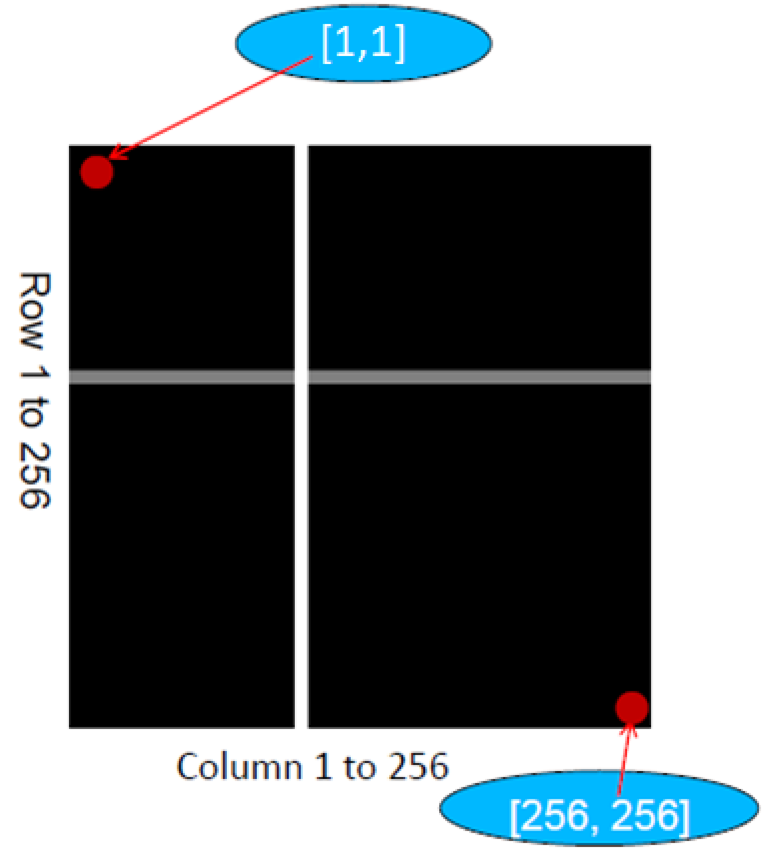

```
230400
uint8
(240, 320, 3)
```

# Python Image Processing

- How to create a matrix (or image)? Density picture:

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

row = 256
col = 256
img = np.zeros((row,col))
img[100:105, :] = 0.5
img[:, 100:105] = 1
plt.figure(figsize=(10,4))
plt.imshow(img)
plt.show()
```



# Python Image Processing

- Binary image

```
import cv2
import numpy as np

height = 512
width = 512
img = np.random.randint(255, size=(height, width, 1), dtype=np.uint8)

cv2.imshow('Binary',img)
```

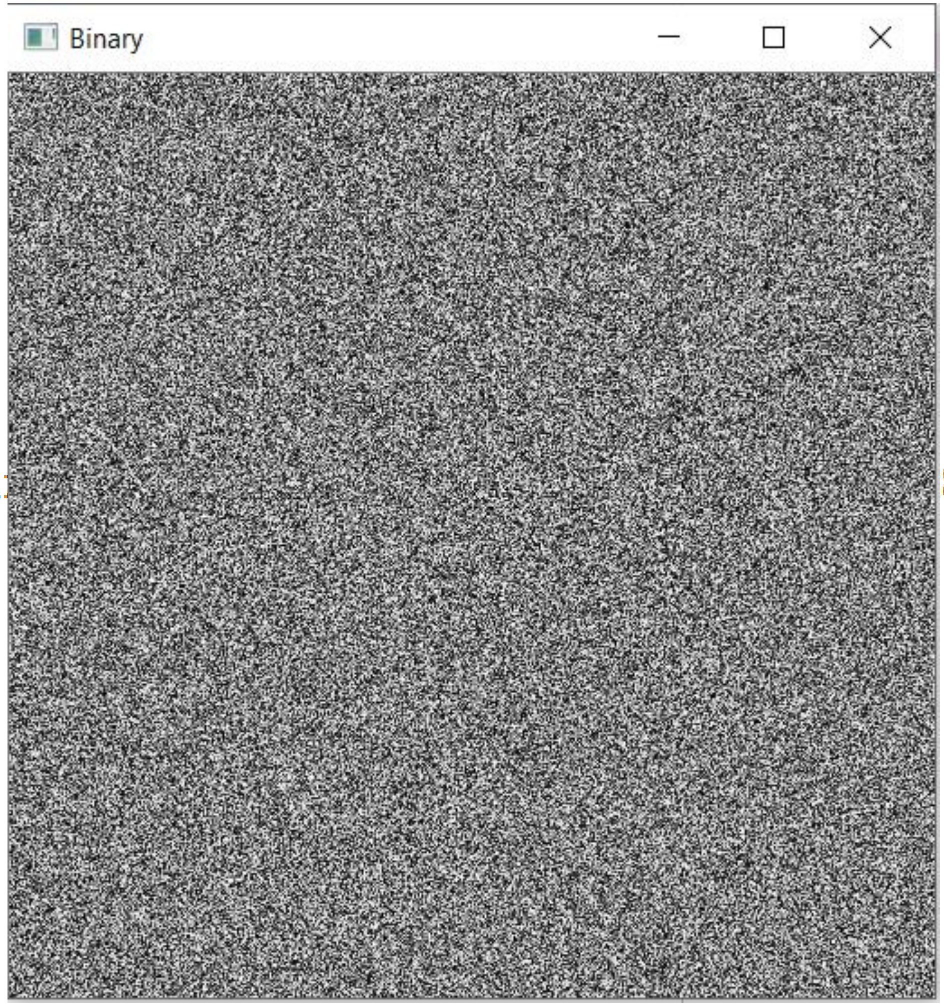
# Python Image Processing

- Binary image

```
import cv2
import numpy as np

height = 512
width = 512
img = np.random.randint(255, si                                   8)

cv2.imshow('Binary',img)
```



# Python Image Processing

## ► Indexing in vectors

>> If the vector v = [1, 3, 5, 7, 9] is typed and push enter

v =

```
[1, 3, 5, 7, 9]
```

>> If value=v[2] is typed and push enter, the second element of vector v is assigned to the variable value:

value =

**5**

A row vector is returned with the transpose function to a column vector:

>> w = np.transpose(v)

w =

1
3
5
7
9

# Python Image Processing

- Indexing in vectors

>> If value = v[1:3] is typed and hit enter, the elements of vector v from index 1 to index 3 are taken:

value =

[ **3, 5 ]**

If you want to take the vector v from the 2nd to the 4th element:

>> value = v[2:4] must be written. Result:

value =

[ **5,7 ]**

If you want to get all the elements from the 3rd element to the end of the vector v:

>> value = v[3:] must be written. Result:

value =

**[ 7, 9 ]**

# Python Image Processing

- Indexing in vectors

When all elements of the vector are requested, starting from the first element and jumping one at a time to the end, the following result is obtained with the following command:>> value = `v[0::2]`

value =

**[1, 5, 9]**

>> v[::-2] command reverses the elements of the vector by jumping from the last element to the first element. Result:

value =

**[9, 5, 1]**

# Python Image Processing

- Indexing in matrices

```python
A = [[1, 4, 5],   # iç içe liste
     [-5, 8, 9],
     [6, 8, 10],
     [0, 2, 38]]


print("A =", A)
print("A[1] =", A[1])       # 2.satır
print("A[1][2] =", A[1][2])   # 2.satırın 3.elemanı
print("A[0][-1] =", A[0][-1])  # İlk satırın sonuncu elemanı
```

```
A = [[1, 4, 5], [-5, 8, 9], [6, 8, 10], [0, 2, 38]]
A[1] = [-5, 8, 9]
A[1][2] = 9
A[0][-1] = 5
```

# Python Image Processing

- Indexing in matrices
  - Matrix can also be created with the numpy library.

```
import numpy as np


a = np.array([[1, 4, 5],    # numpy array
              [-5, 8, 9],
              [6, 8, 10],
              [0, 2, 38]])


print("type:", type(a))
print("a =","\n",a, "\n")
print("a[1] =", a[1])       # 2.satır
print("a[1][2] =", a[1][2])   # 2.satırın 3.elemanı
print("a[0][-1] =", a[0][-1])  # İlk satırın sonuncu elemanı
```

```
type: <class 'numpy.ndarray'>
a =
 [[ 1  4  5]
 [-5  8  9]
 [ 6  8 10]
 [ 0  2 38]]

a[1] = [-5  8  9]
a[1][2] = 9
a[0][-1] = 5
```

# Python Image Processing

- Indexing in matrices

```
import numpy as np
C = np.array([[1, 1, 2], [3, 5, 3], [5, 6, 9]])
C[:,2]

Output: array([2, 3, 9])
```

- The ( : ) operator selects elements in blocks. In the above code it means all rows but only 2nd column. It will only give us the 2nd column elements.
- >> C[ : , 2 ] = 1

Makes only the elements of the second column of the C matrix 1 in all rows.

# Python Image Processing

- Indexing in matrices

  - Example question: How can we make a matrix B whose third column is 0 but all other columns are equal to A??

    Cevap: >> B = A ;

    >> B[: , 2] = 0 ;

  - Sample ; >> sum ( C[ : ] ) and sum(C) expressions give the same result. Both give the sum of the elements in each column itself.

```
C = np.array([[1, 1, 2], [3, 5, 3], [5, 6, 9]])
s = sum ( C[ : ] )
print(s)
a = sum(C)
print(a)
```

```
Output:

        [ 9 12 14] [ 9 12 14]
```

# Python Image Processing

- Image Operations
  - RGB image to gray image
  - Image resize
  - Image cropping
  - Rotate image
  - Image histogram
  - Image histogram equalization
  - Convolution

# References


- Sayısal Görüntü İşleme, Palme Publishing, Third Press Trans. (*Orj: R.C. Gonzalez and R.E. Woods: "Digital Image Processing", Prentice Hall, 3rd edition, 2008*).
- "Digital Image Processing Using Matlab", Gonzalez & Richard E. Woods, Steven L. Eddins, Gatesmark Publishing, 2009
- Lecture Notes, MATLAB for Image Processing, CS638-1 TA: Tuo Wang
- Lecture Notes, BIL717-Image Processing, Erkut Erdem
- Lecture Notes, EBM537-Image Processing, F.Karabiber
- https://docs.opencv.org/
- Bekir Aksoy, Python ile İmgeden Veriye Görüntü İşleme ve Uygulamaları, Nobel Akademik Yayıncılık