High Impact Skills Development Program

in Artificial Intelligence, Data Science, and Blockchain

Module: [Computer Vision]

Lecture 1: [Introduction to Computer Vision]

Instructor: [Dr. Rabia Irfan]

[Assistant Professor], SEECS, NUST

Module Objectives

- To understand the basics of Computer Vision and Image Processing
- To discuss practical applications of Computer Vision
- To learn about Convolutional Neural Networks (CNNs)
- To understand Transfer Learning and Model Optimization Concepts
- To gain hands-on experience with relevant Python libraries

Agenda

- What is Computer Vision?
- Applications of Computer Vision
- Computer Vision Tasks and Trends
- Fundamentals of Digital Image Processing
- What is a Digital Image?
- Image Manipulations and Geometric Transformations

Introduction to Computer Vision

Lecture 1- A

What is Computer Vision?

- Computer Vision provides the ability to computers to see and understand images

Image credit: Manfred-Suzanne DANEGGER/Gamma-Rapho via Getty Images

What is Computer Vision?

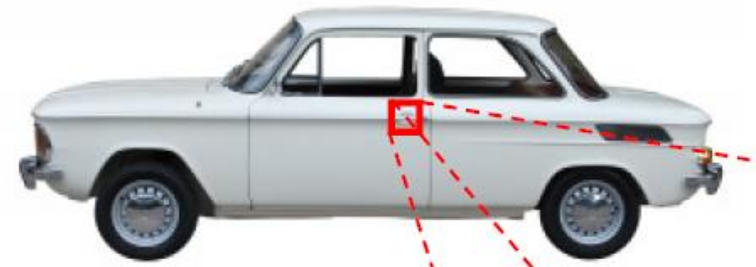
Making a computer understand:

- What animal is this?
- How many lions are there?
- What are they doing?

Image credit: Manfred-Suzanne DANEGGER/Gamma-Rapho via Getty Images

What is Computer Vision?

You see this:



But the camera sees this:

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

Image credit: Slides by Andrew Ng

Applications of Computer Vision

- Automotive
- Waste Management
- Manufacturing
- Pharmaceutical
- Healthcare
- Customer Service
- Energy & Utilities
- Insurance
- Security & Surveillance
- Entertainment

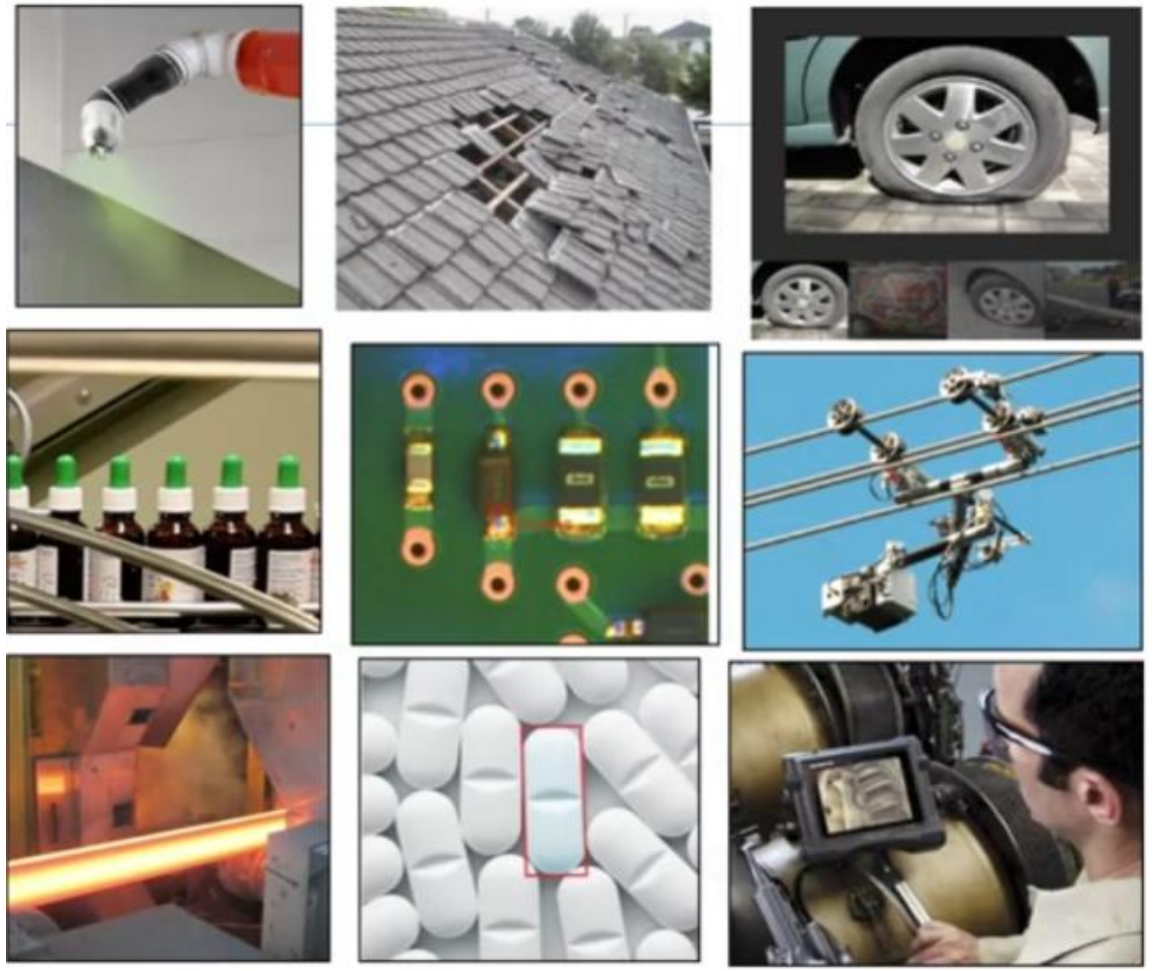


Image credit: coursera

Applications of Computer Vision

- ADNOC (Abu Dhabi National Oil Company)
- Uses Computer Vision with IBM Watson to classify 25,000 rock images per day

https://www.ibm.com/case-studies/abu-dhabi-national-oil-company-adnoc

Why use Computer Vision?

- Knockri – A Canadian-based startup
- Soft skills assessment tool to help large companies shortlist the candidates

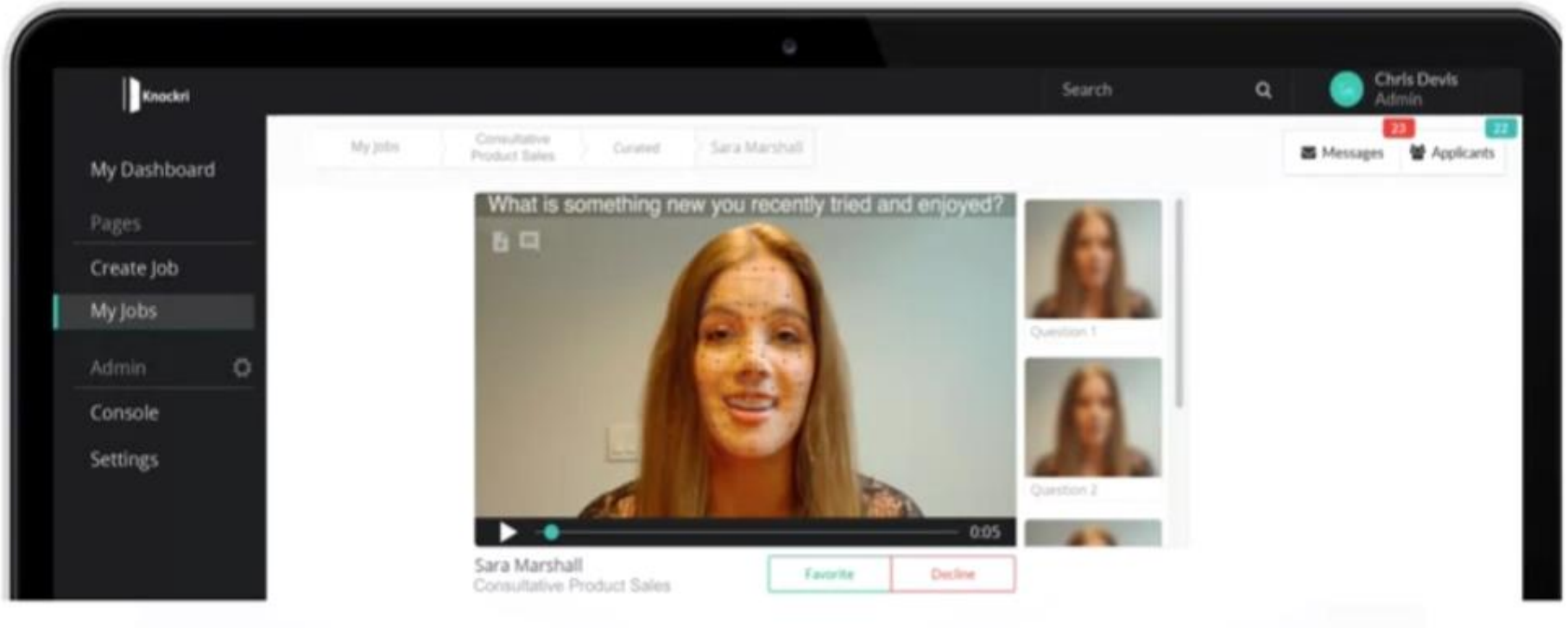


Image credit: coursera

Applications of Computer Vision

- Rust and Damage Detection

Image credit: coursera

Applications of Computer Vision

- Rust and Damage Detection

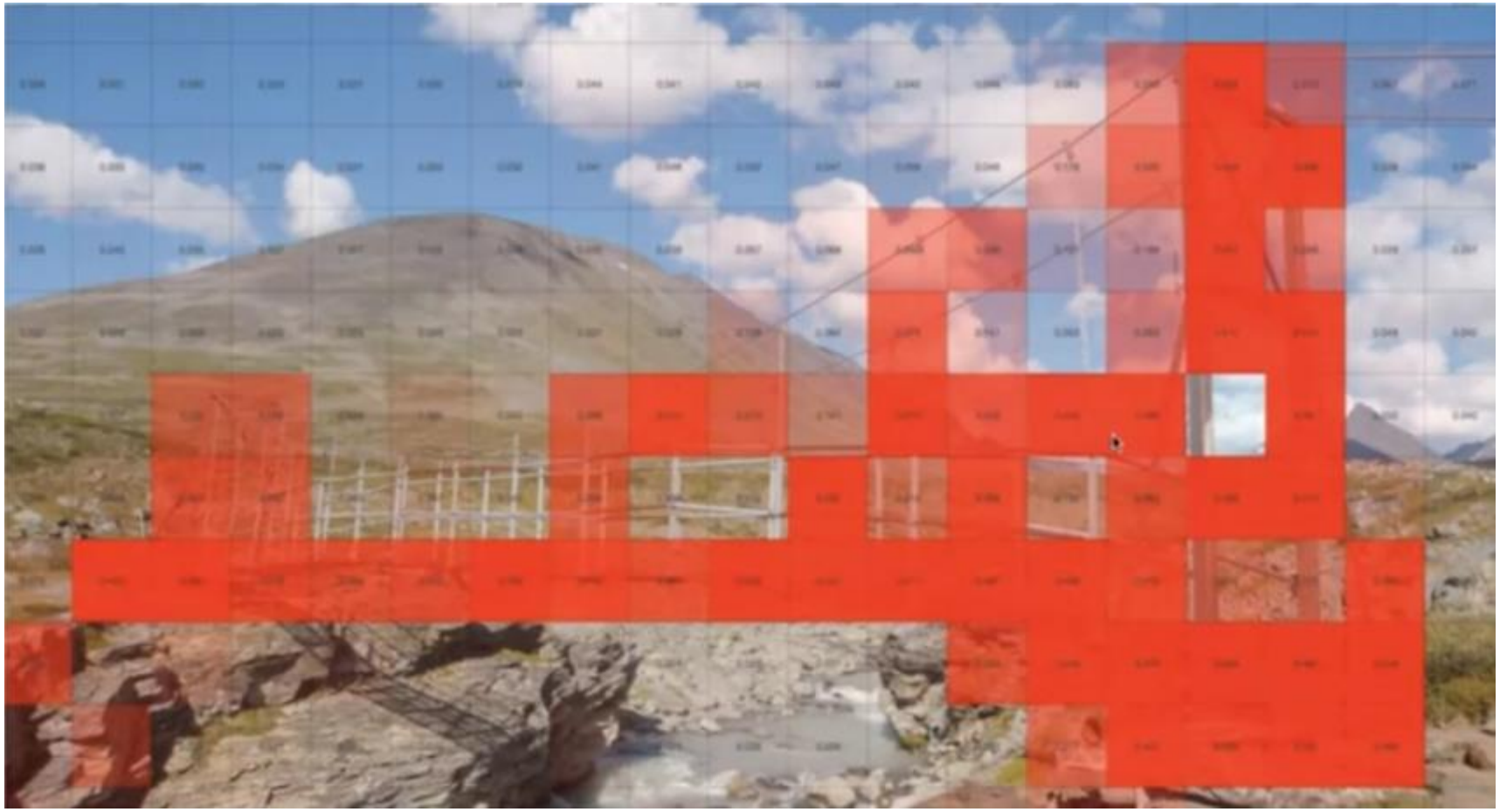


Image credit: coursera

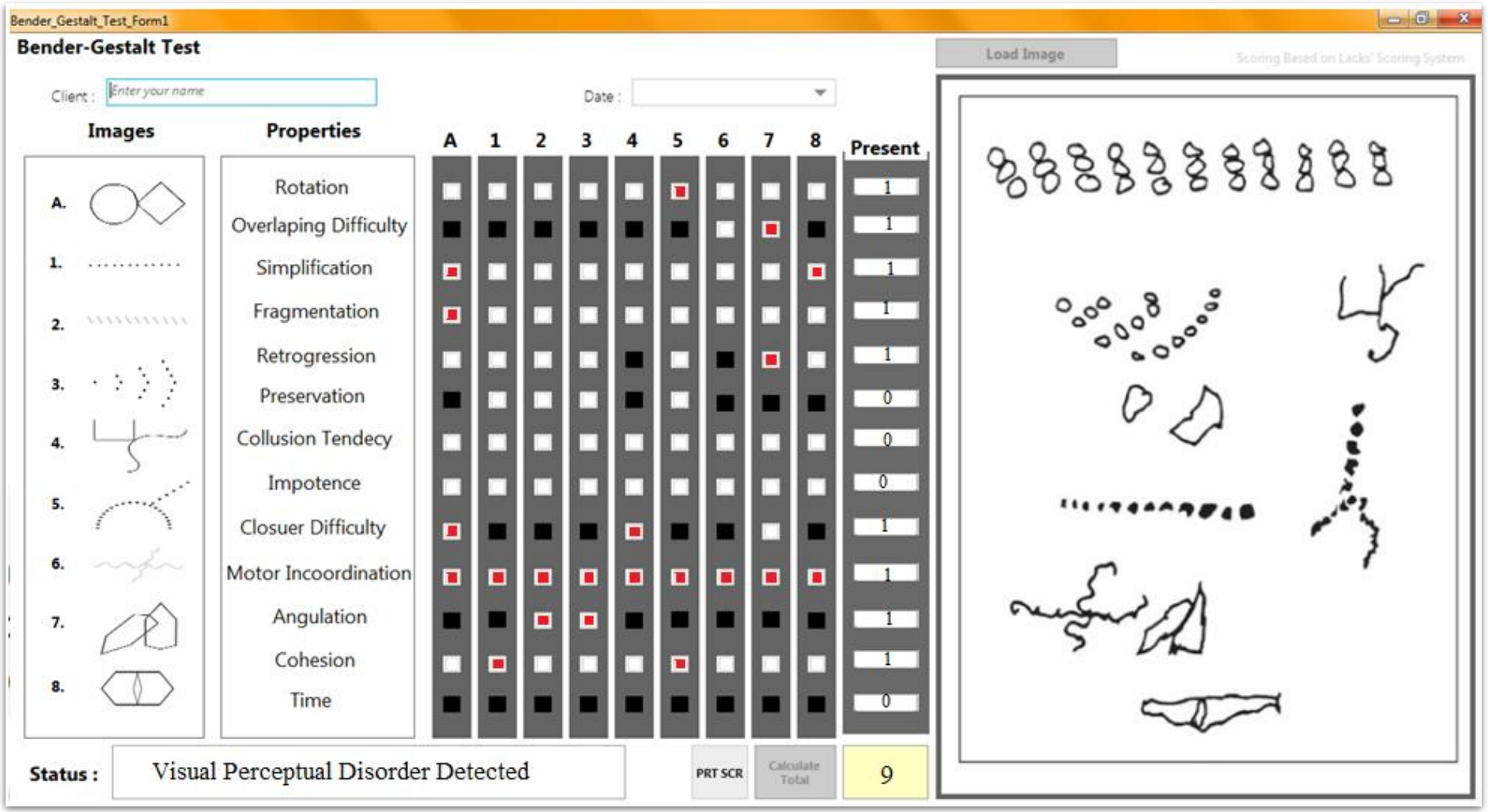
Applications of Computer Vision

- Security & Surveillance

Image credit: coursera

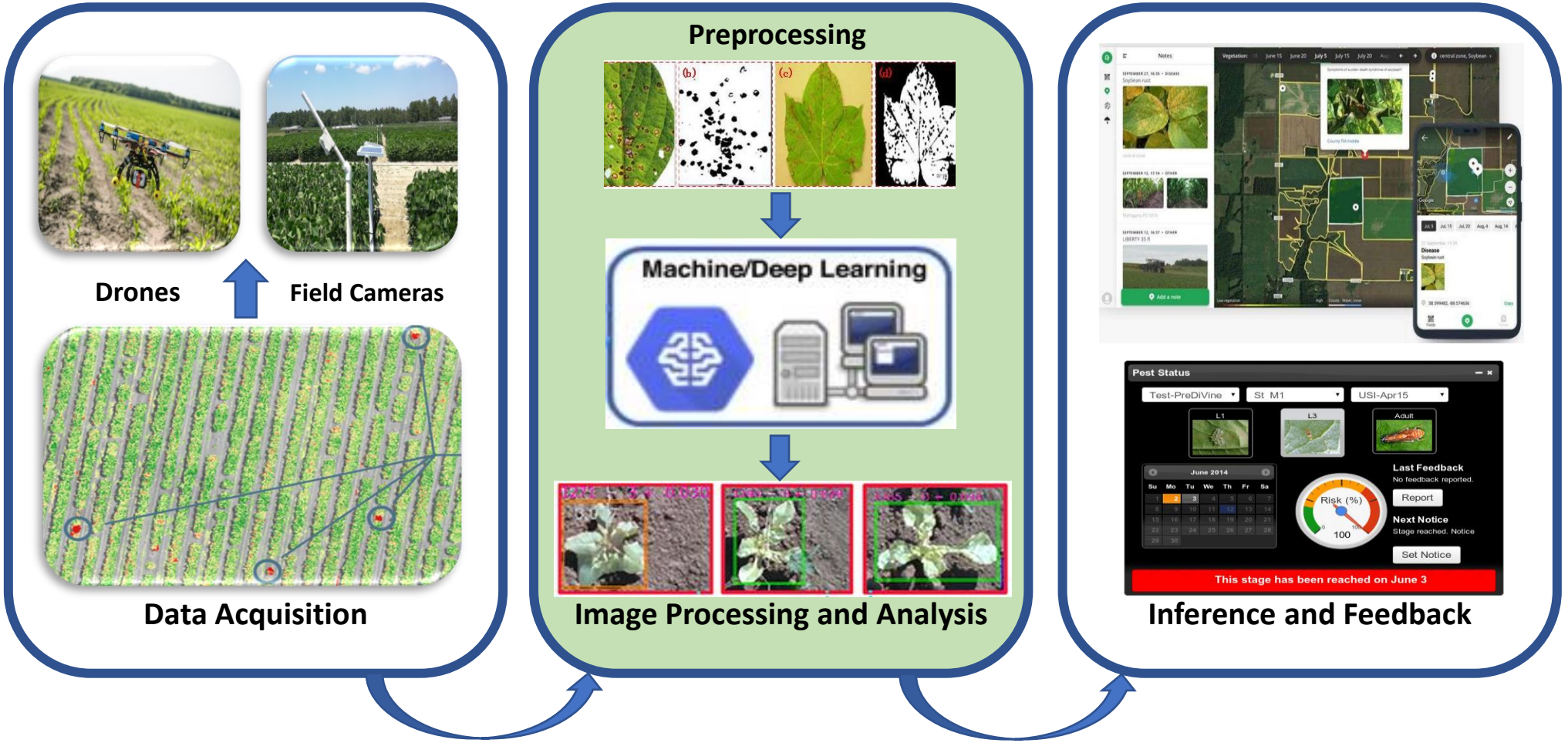
Applications of Computer Vision

- Computer Vision in Healthcare

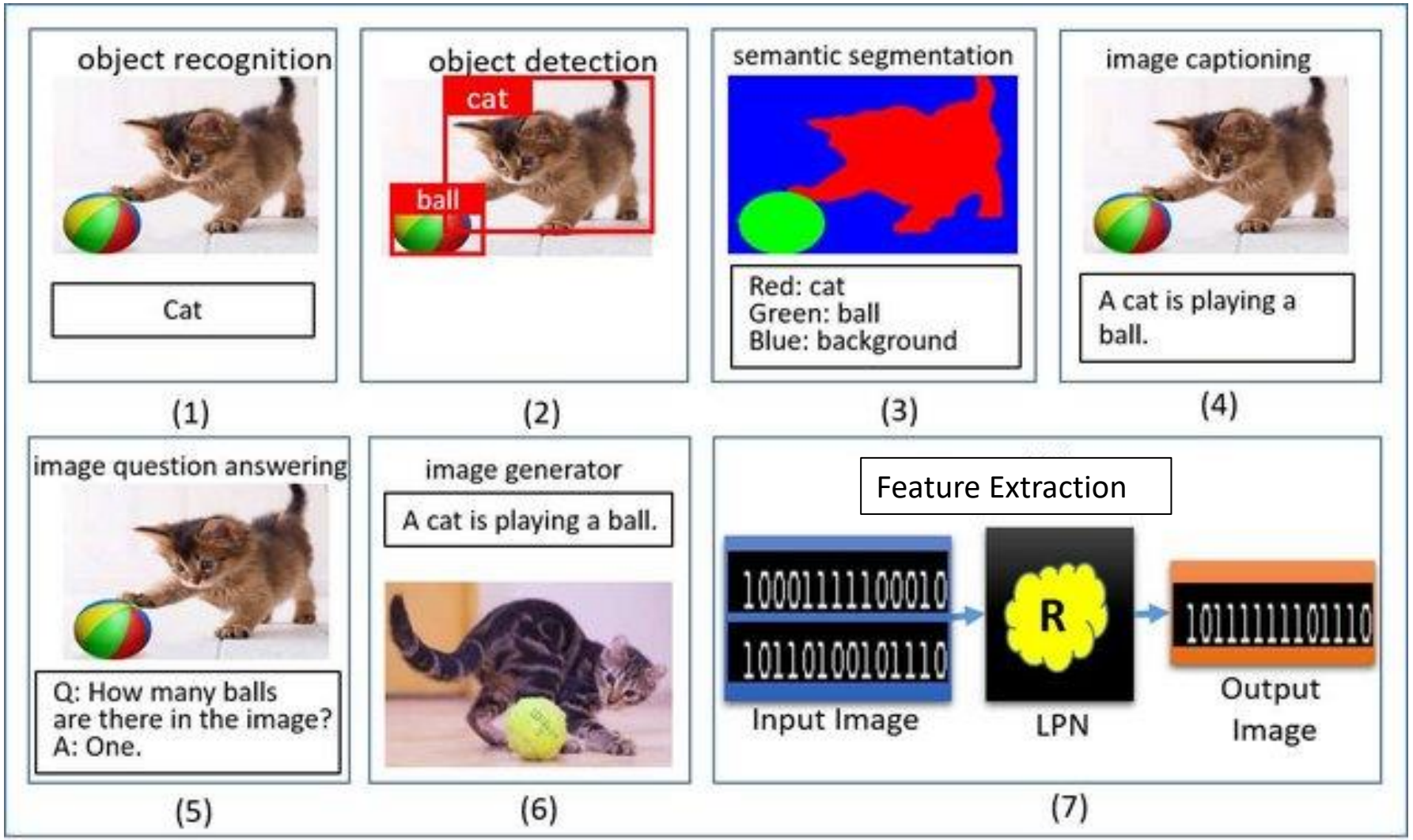


Applications of Computer Vision

- Computer Vision in Agriculture



Traditional Computer Vision Tasks



Guo, Q., Qian, Y., Liang, X. *et al.* Logic could be learned from images. *Int. J. Mach. Learn. & Cyber.* **12**, 3397–3414 (2021). https://doi.org/10.1007/s13042-021-01366-w

Advances in Computer Vision

- Image-to-Image Translation

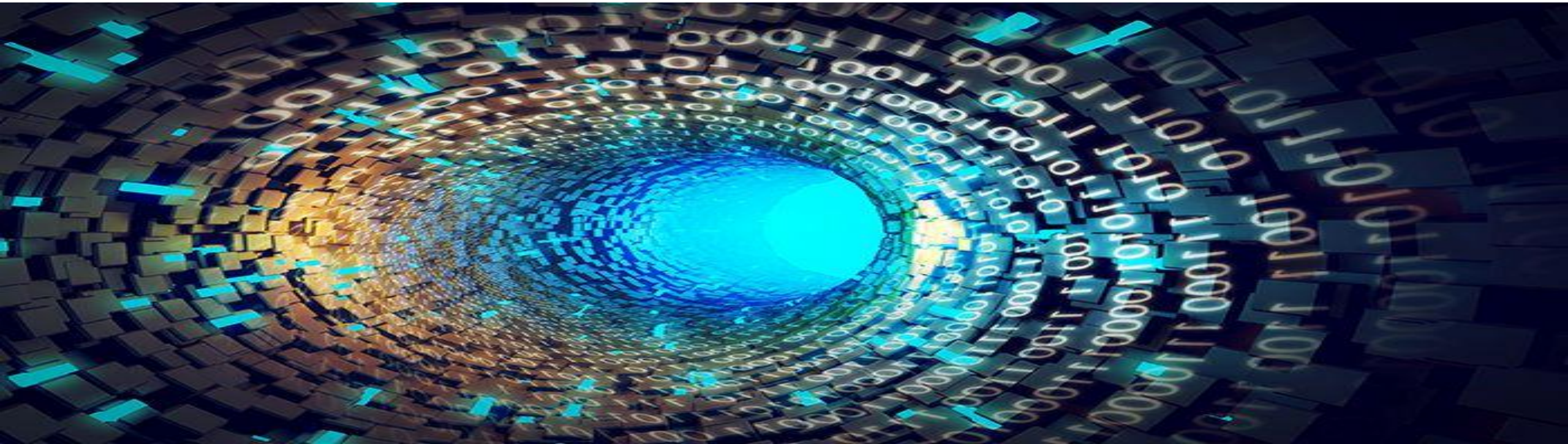
Zhu et al. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. ICCV. https://arxiv.org/pdf/1703.10593v6.pdf

Advances in Computer Vision

- Deep Fakes

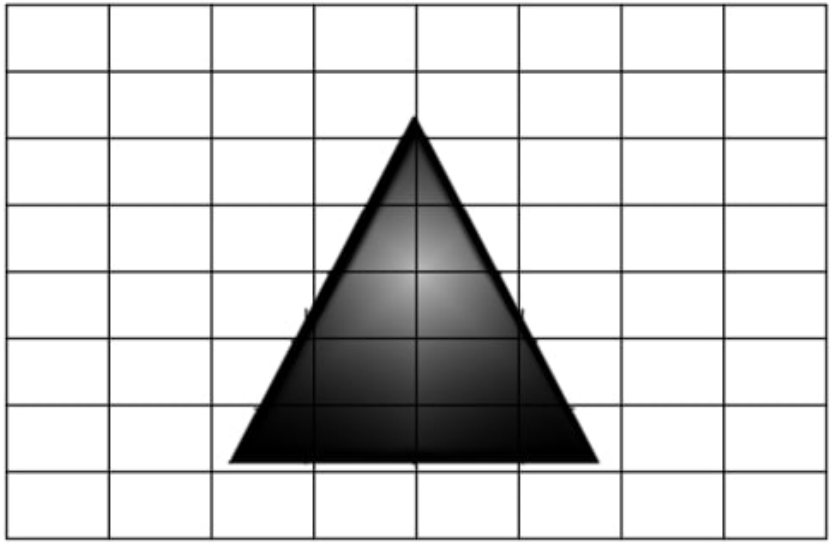
Fundamentals of Digital Image Processing

Lecture 1 - B

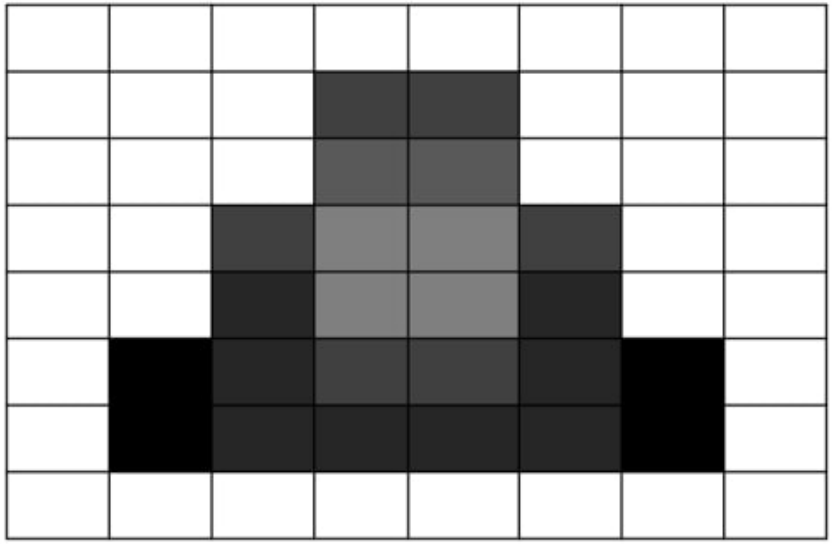


What is a Digital Image?

Digital Image is an approximation of a real world scene



Continuous image projected onto a sensor array.



Result of image sampling and quantization.

Digital Image Formation

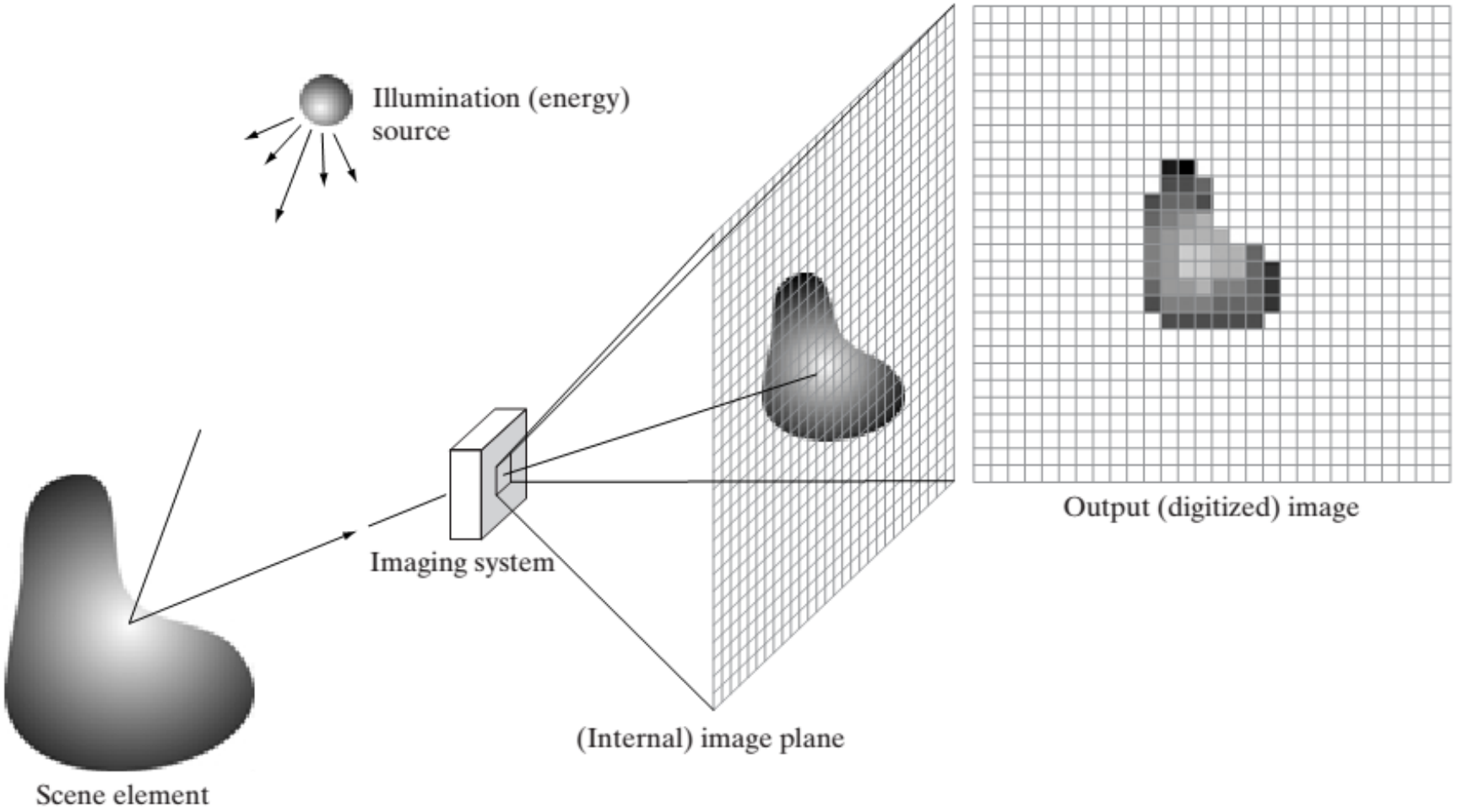
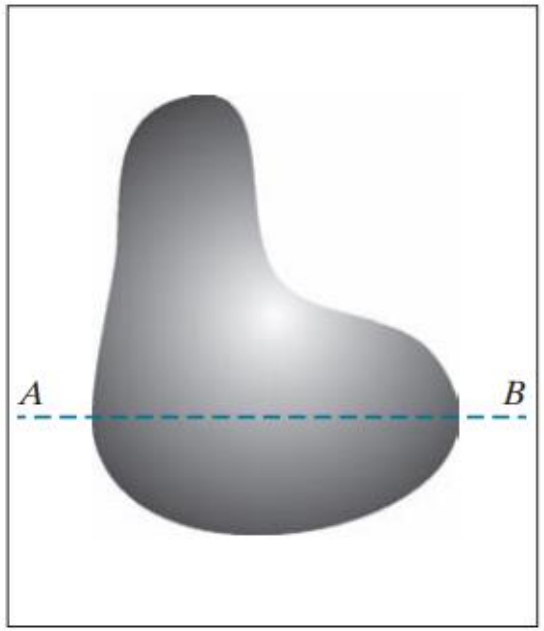
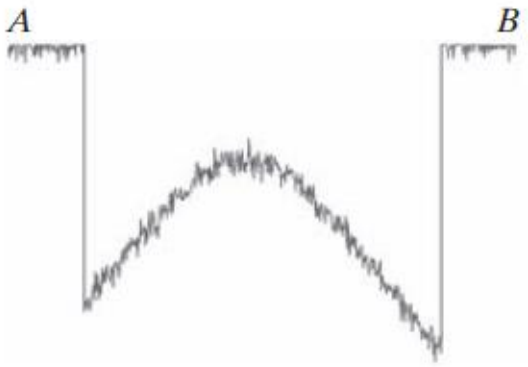
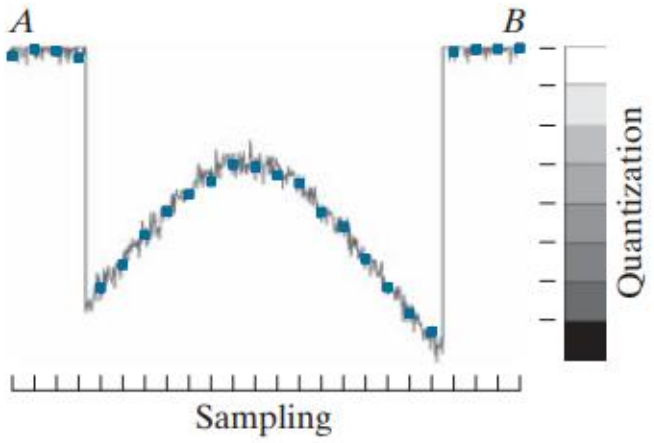


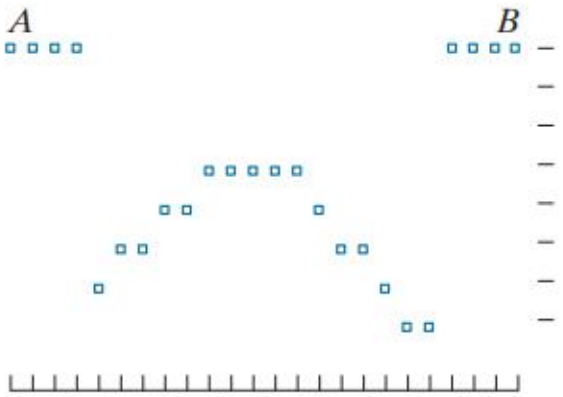
Image credit: Digital Image Processing by Gonzalez

Continuous to Digital









Sampling

- Sampling is a **spatial resolution** of the digital image.
- The rate of sampling determines the **quality** of the digitized image.

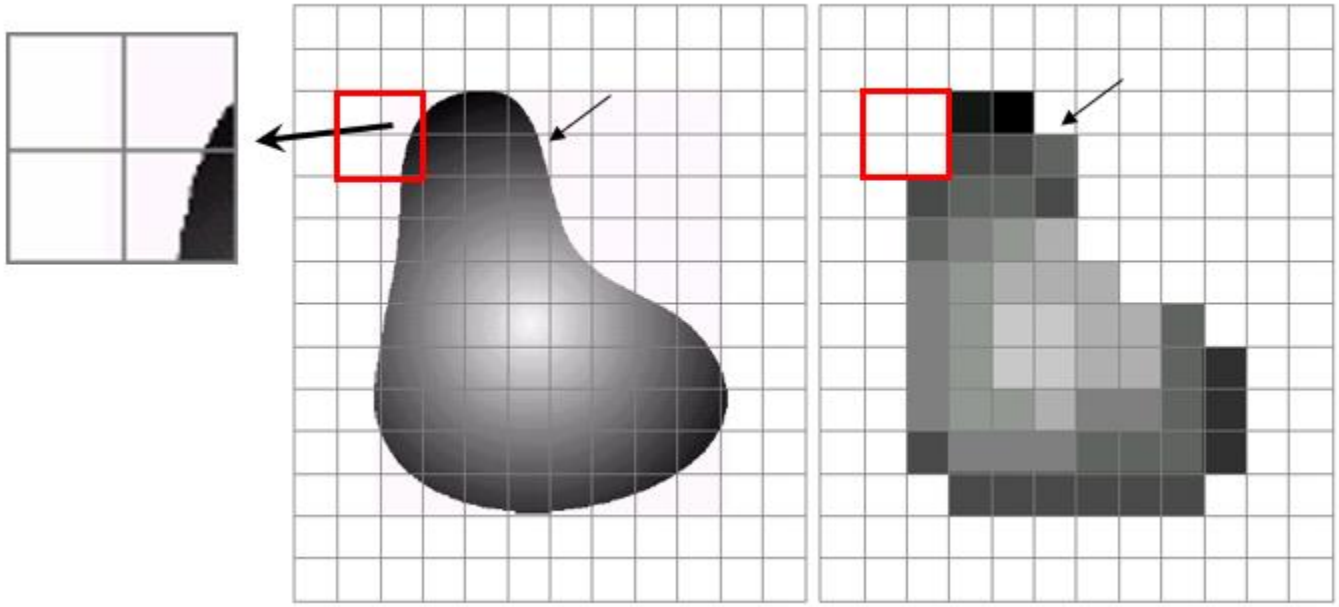
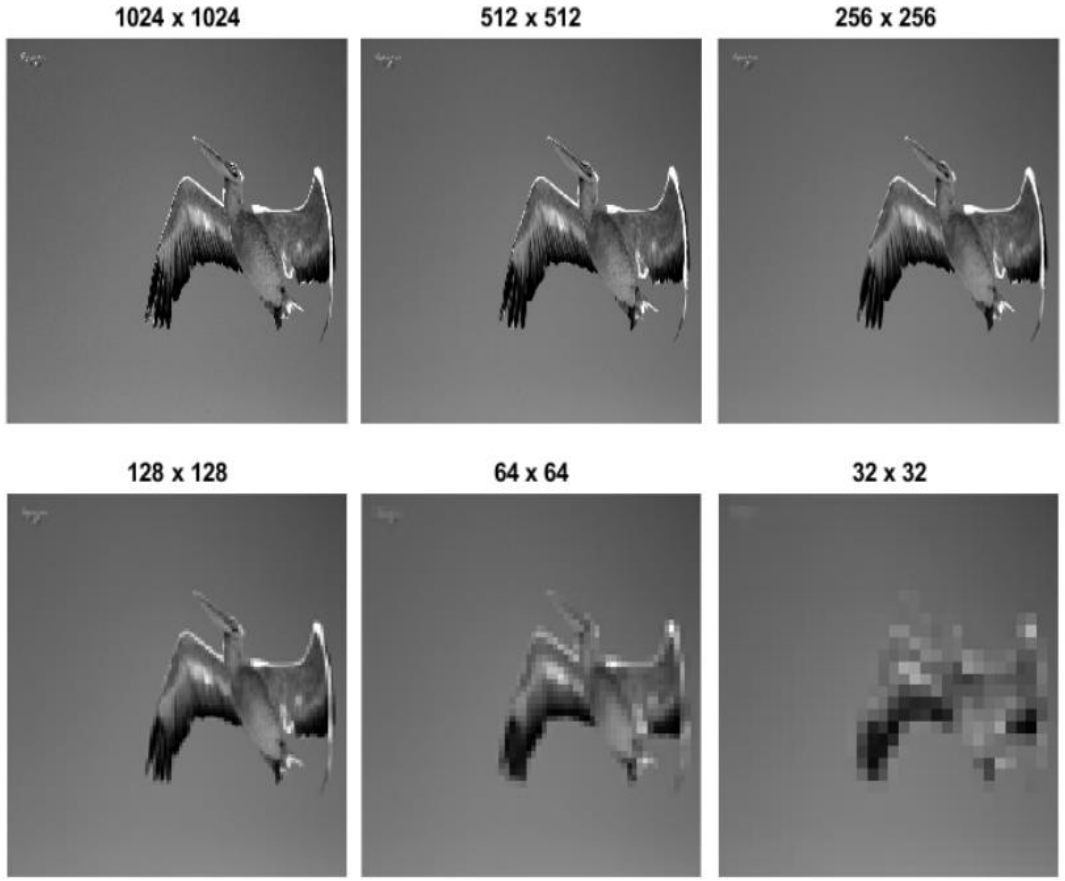


Image credit: Digital Image Processing by Gonzalez

Spatial Resolution

- The spatial resolution of an image is determined by how sampling was carried out.
- There are 3 measures which we see often relating to Image Size/Resolution
  a. **Pixel count - e.g., 3000x2000 pixels**
  b. **Physical size - e.g., 8" x 10"**
  c. **Resolution - e.g., 240 pixels per inch (PPI)**

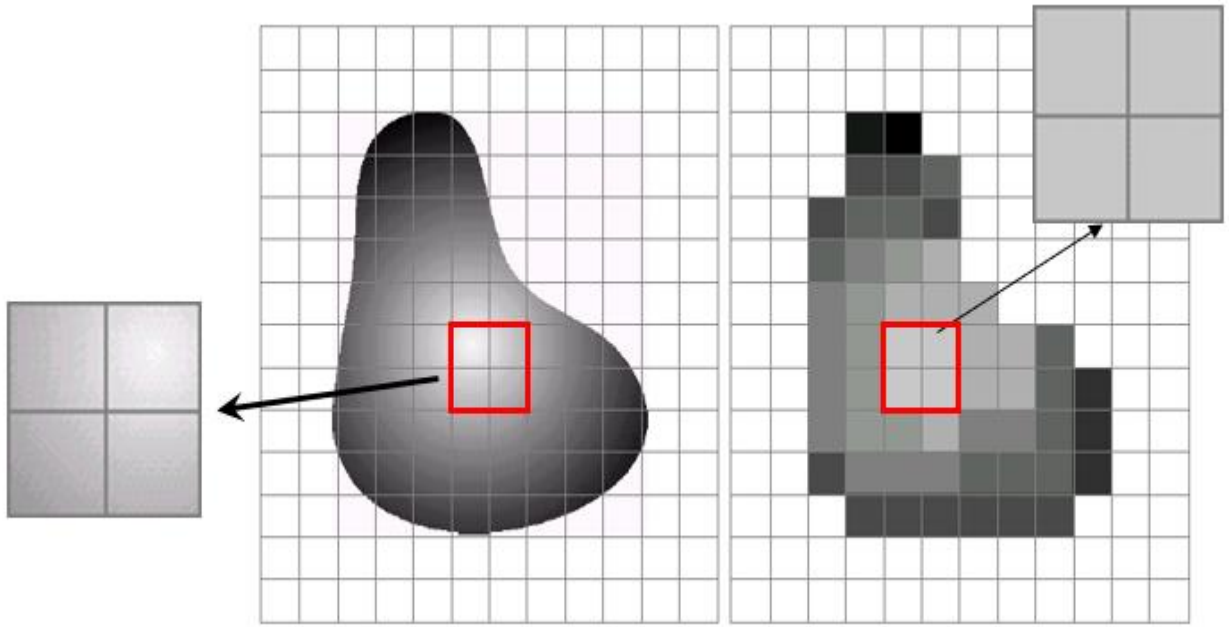


Quantization

- The transition of the continuous values from the image function to its digital equivalent is called quantization.
- Quantization is the number **of grey levels** in the digital image.
- It is related to the **intensity** values of the image.
- 8-bit quantization: 2^8 =256 gray levels

  (0: black, 255: white)
- 1-bit quantization: 2 gray levels

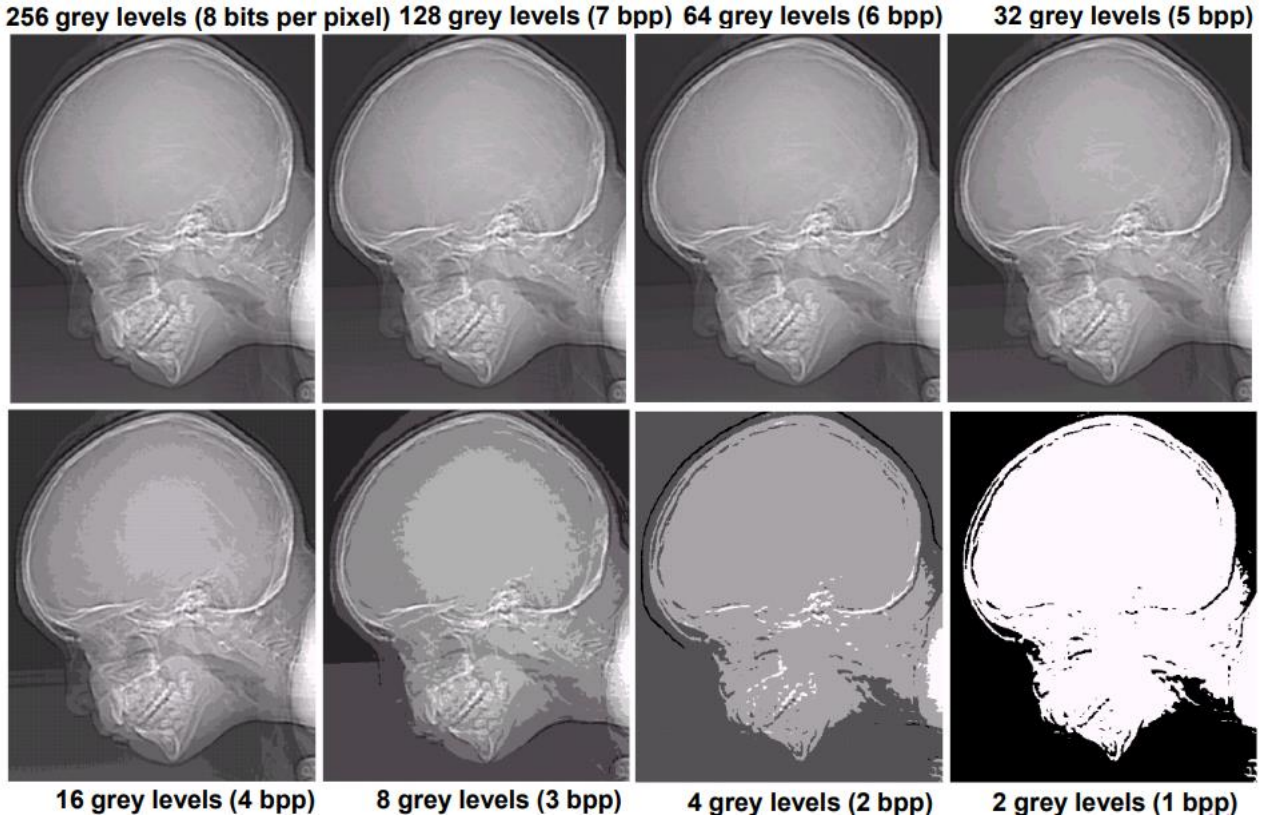
  (0: black, 1: white) – binary



Number of Bits	Number of Intensity Levels	Examples
1	2	0, 1
2	4	00, 01, 10, 11
4	16	0000, 0101, 1111
8	256	00110011, 01010101
16	65,536	1010101010101010

Intensity Resolution

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



Resolution: How Much is Enough?

Image Contains Information

Image with low level of details

Image with medium level of details

Image with highest level of details

Image Representations

- Gray scale Images

Columns

rows

$$\begin{bmatrix} 2 & 3 & .. & 40 \\ : & : & .. & 50 \\ 5 & 34 & .. & 24 \end{bmatrix}$$

Image Representations

- Colored (RGB) Images

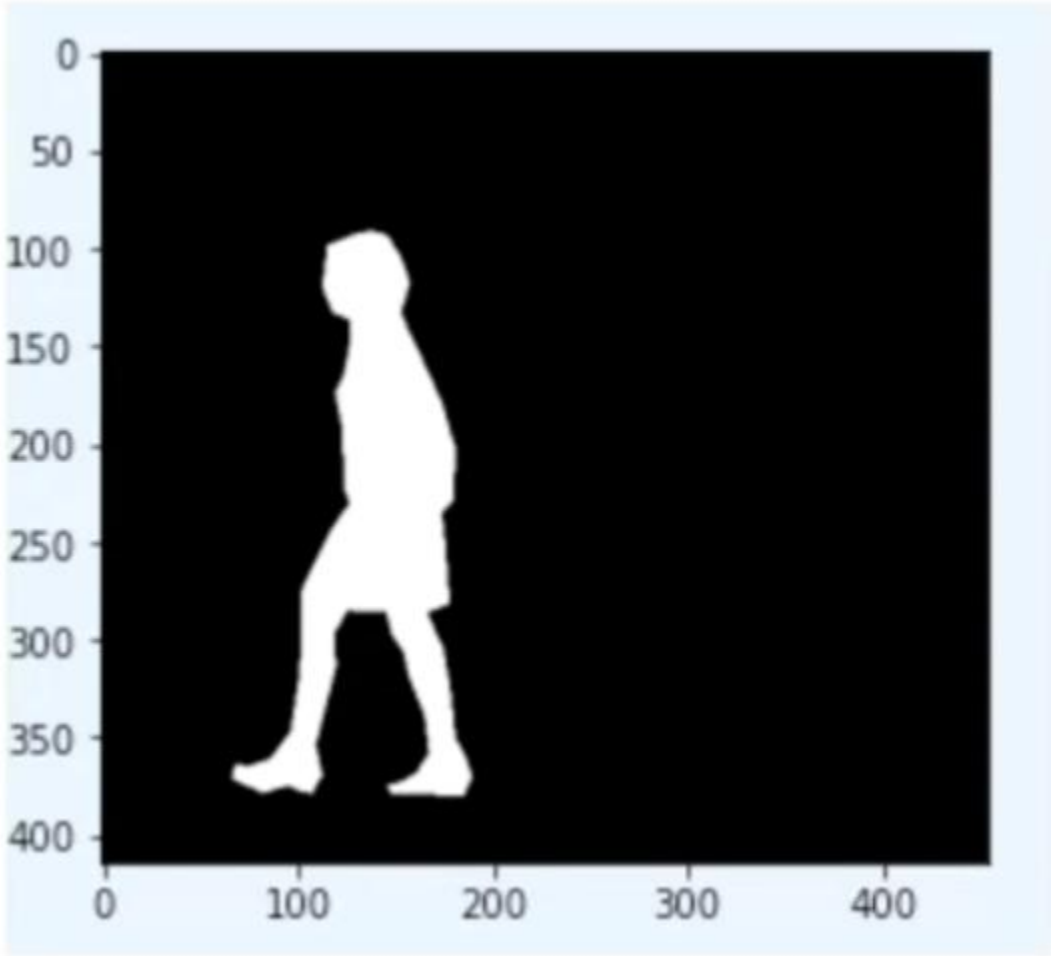
$$\begin{bmatrix} 9 & 3 & .. & 40 \\ : & : & .. & 50 \\ 5 & 34 & .. & 24 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 3 & .. & 40 \\ : & : & .. & 50 \\ 5 & 34 & .. & 24 \end{bmatrix}$$

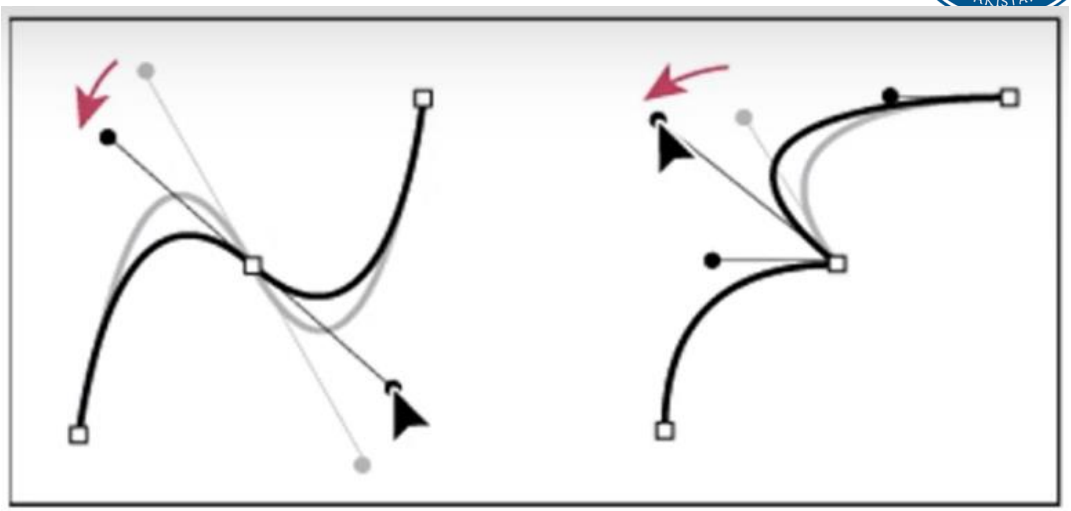
$$\begin{bmatrix} 4 & 3 & .. & 40 \\ : & : & .. & 50 \\ 5 & 34 & .. & 24 \end{bmatrix}$$

Image Representations

- Black and White (Binary) Image



Classification – Raster vs. Vector (Space vs. Quality)



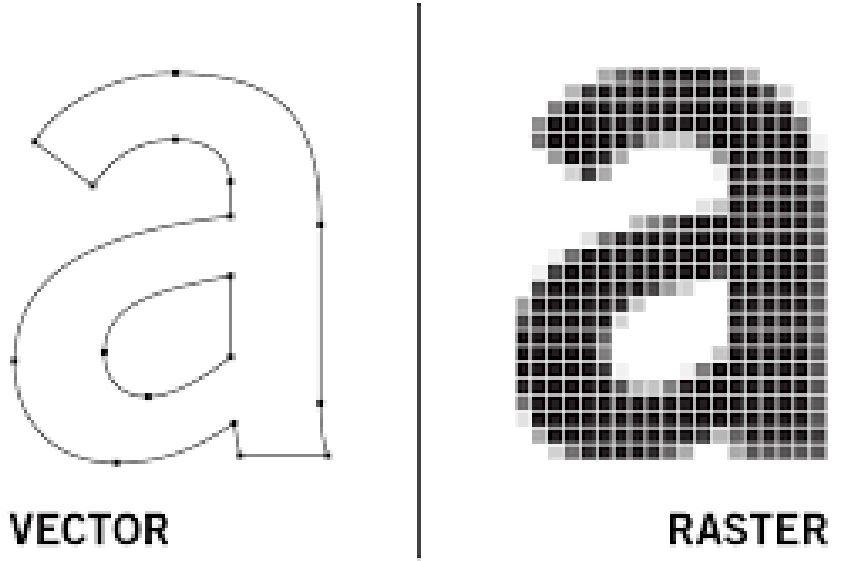


Image Formats

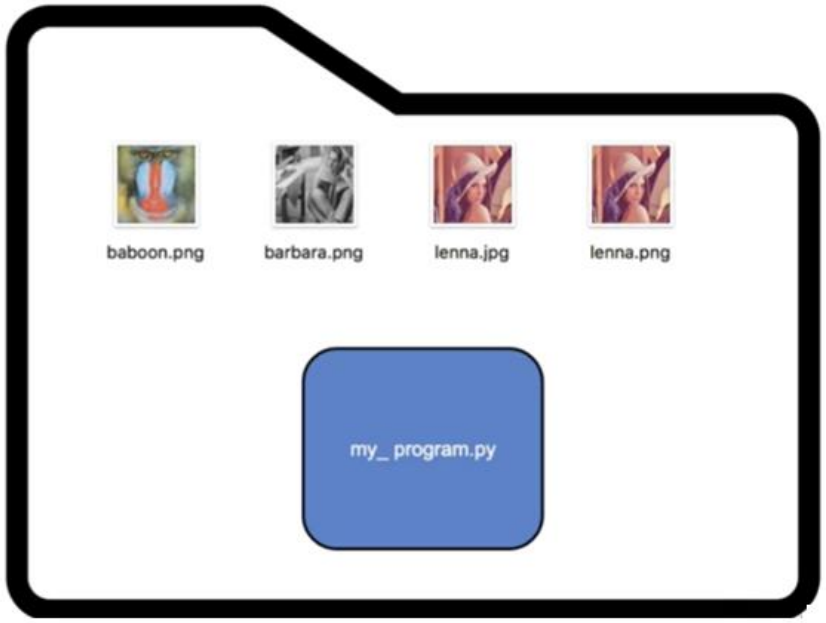
- Image Formats
  - JPEG (Raster)
  - PNG (Raster)
  - GIF (Raster)
  - BMP (Raster)
  - SVG (Vector)

Storing Images

Lossy vs. Lossless (Compression)

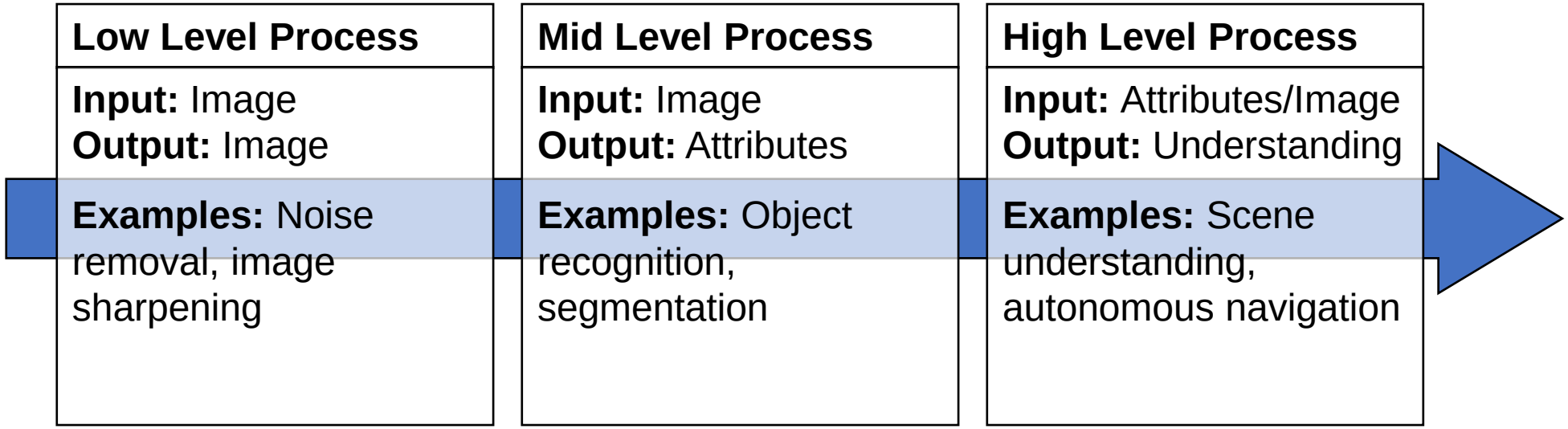
Color Range

Pixels vs. Vector

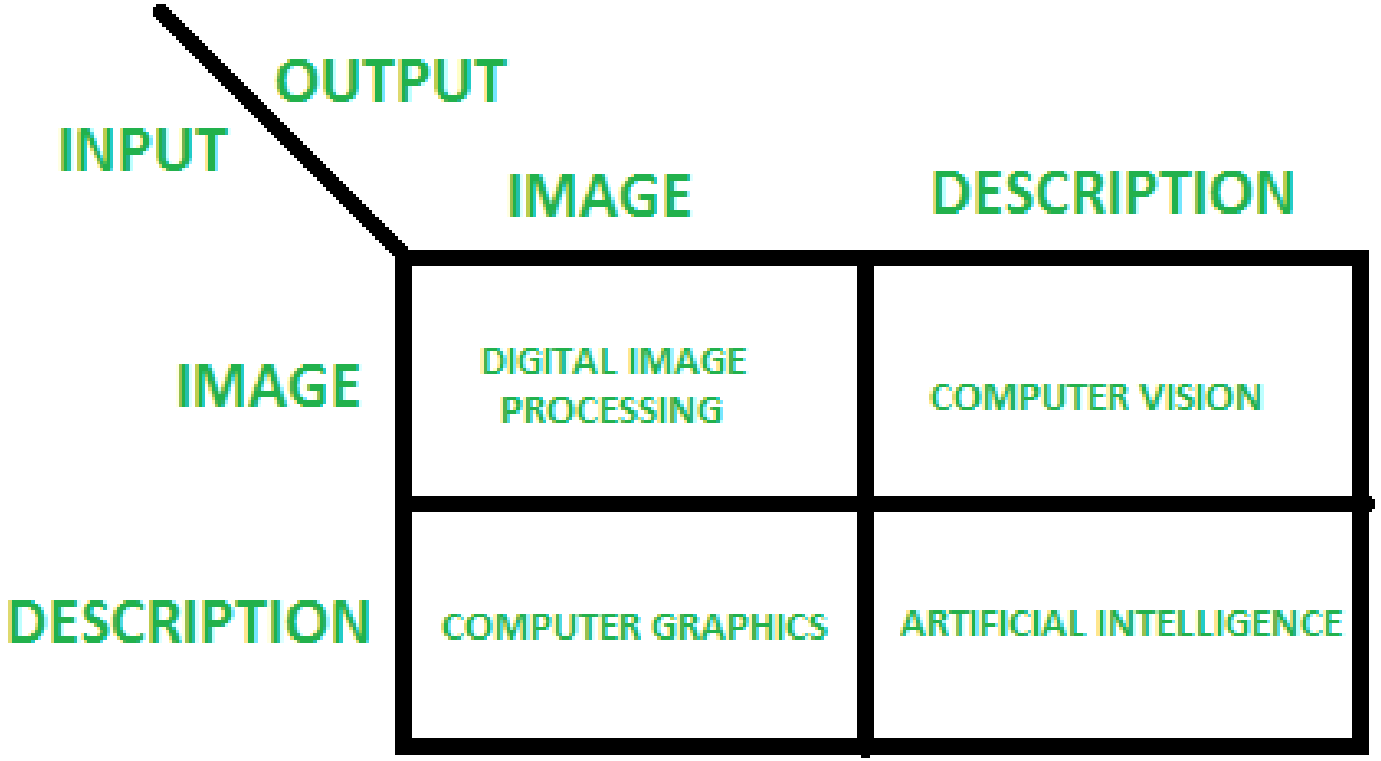


Relation of DIP and CV

- Digital Image Processing is a method to perform some operations on the image in order to get an **enhanced image** or to **extract some useful information from it.**



Relation of DIP and CV

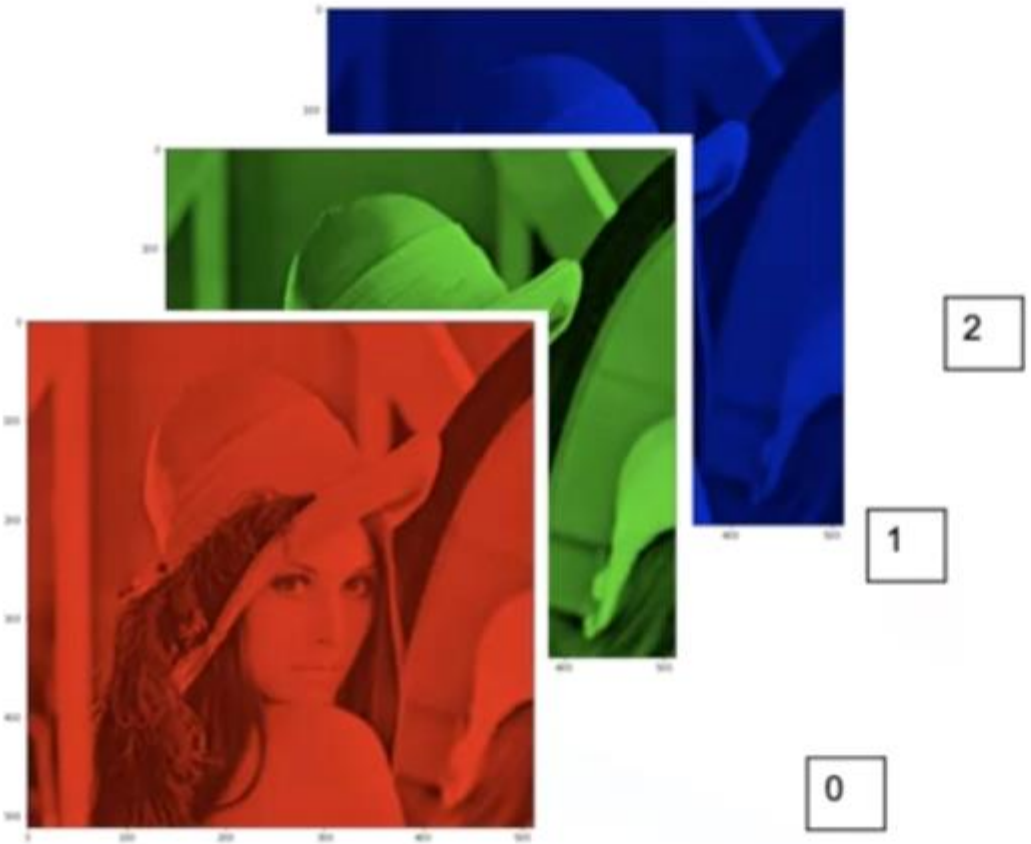


Python Libraries for DIP and CV

- Pillow
- OpenCV

OpenCV	PIL (Python Imaging Library)
- Open Source Computer Vision	- Image processing package exclusively for Python
- Supports Python, C++, Java	- A project named Pillow is forked to the original PIL library for its use in Python3.x and above
- Processes images and videos for feature extraction	
- Reads the images in BGR format by default	- Reads the images in RGB format by default

PIL vs OpenCV



Working with PILLOW (PIL)

```
from PIL import Image

image = Image.open(my_image)

import matplotlib.pyplot as plt
plt.imshow(image)

image.format:PNG

image.size:(512, 512)

image.mode:RGB
```

PIL.PngImagePlugin.PngImageFile

Working with PILLOW (PIL)

```
from PIL import ImageOps

image_gray = ImageOps.grayscale(image)

 image_gray.mode:L


image_gray.save("lenna.jpg")
```

Working with OpenCV

```
new_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

plt.imshow(new_image )
```

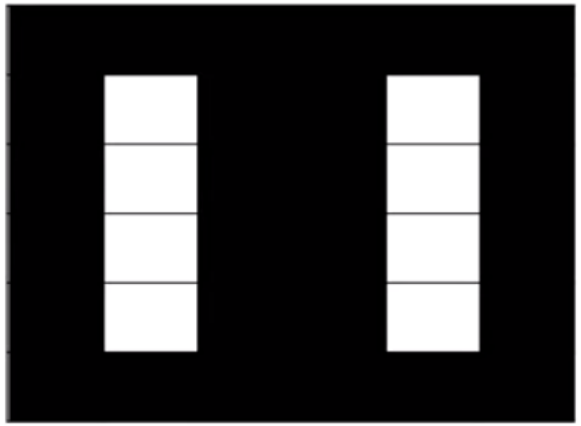
Working with OpenCV

```
image_gray = cv2. cvtColor(image, cv2.COLOR_BGR2GRAY)

plt.imshow(image_gray )

cv2.imwrite('lena_gray_cv.jpg', image_gray)
```

Image Manipulations

I[0,0]	I[0,1]	I[0,2]	I[0,3]	I[0,4]	I[0,5]
I[1,0]	I[1,2]	I[1,2]	I[1,3]	I[1,4]	I[1,5]
I[2,0]	I[2,2]	I[2,2]	I[2,3]	I[2,4]	I[2,5]
I[3,0]	I[3,1]	I[3,2]	I[3,3]	I[3,4]	I[3,5]
I[4,0]	I[4,1]	I[4,2]	I[4,3]	I[4,4]	I[4,5]
I[5,0]	I[5,1]	I[5,2]	I[5,3]	I[5,4]	I[5,5]

I[0,0]	I[1,0]	I[2,0]	I[3,0]	I[4,0]	I[5,0]
I[0,1]	I[1,1]	I[2,1]	I[3,1]	I[4,1]	I[5,1]
I[0,2]	I[1,2]	I[2,2]	I[3,2]	I[4,2]	I[5,2]
I[0,3]	I[1,3]	I[2,3]	I[3,3]	I[4,3]	I[5,3]
I[0,4]	I[1,4]	I[2,4]	I[3,4]	I[4,4]	I[5,4]
I[0,5]	I[1,5]	I[2,5]	I[3,5]	I[4,5]	I[5,5]



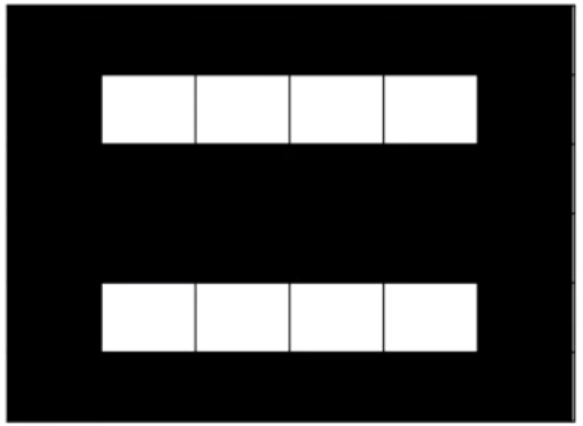


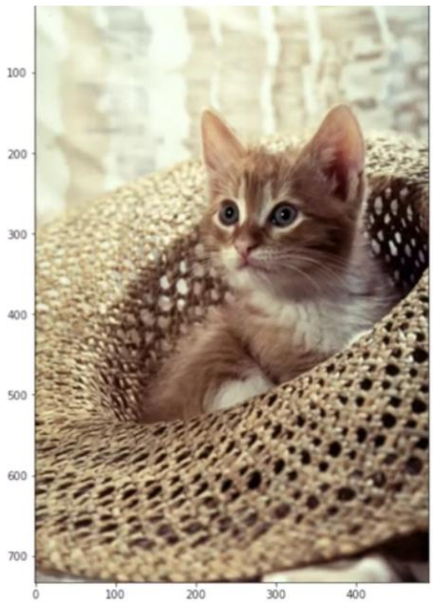
Image Manipulations

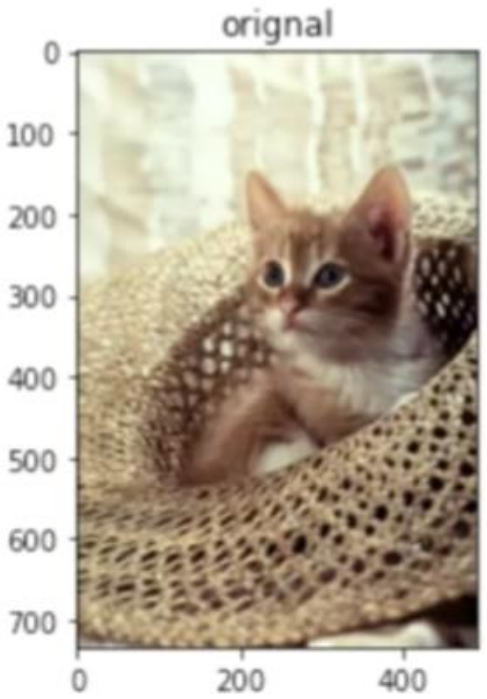
```
from PIL import  ImageOps

im_flip = ImageOps.flip(image)

im_mirror = ImageOps.mirror(image)
```

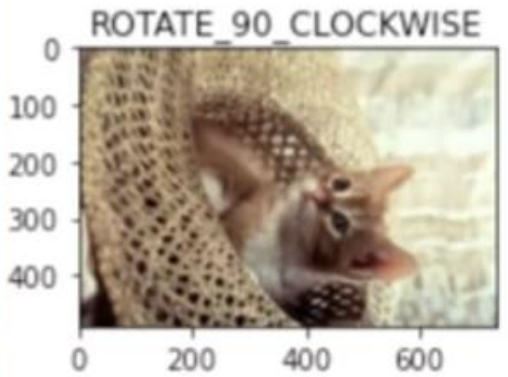
pillow (PIL)

Image Manipulations

```
import cv2

im_flip =  cv2.flip(image,0 )

im_flip=cv2.rotate(image, cv2.ROTATE_90_CLOCKWISE)
```







Manipulating Images

I =

0	I[0,0]	I[0,1]	I[0,2]	I[0,3]	I[0,4]	I[0,5]
1	I[1,0]	I[1, 1]	I[1,2]	I[1,3]	I[1,4]	I[1,5]
2	I[2,0]	I[2, 1]	I[2,2]	I[2,3]	I[2,4]	I[2,5]
3	I[3,0]	I[3,1]	I[3,2]	I[3,3]	I[3,4]	I[3,5]
4	I[4,0]	I[4,1]	I[4,2]	I[4,3]	I[4,4]	I[4,5]
5	I[5,0]	I[5,1]	I[5,2]	I[5,3]	I[5,4]	I[5,5]

0	1	2	3	4	5

Manipulating Images

I[2:5,:]

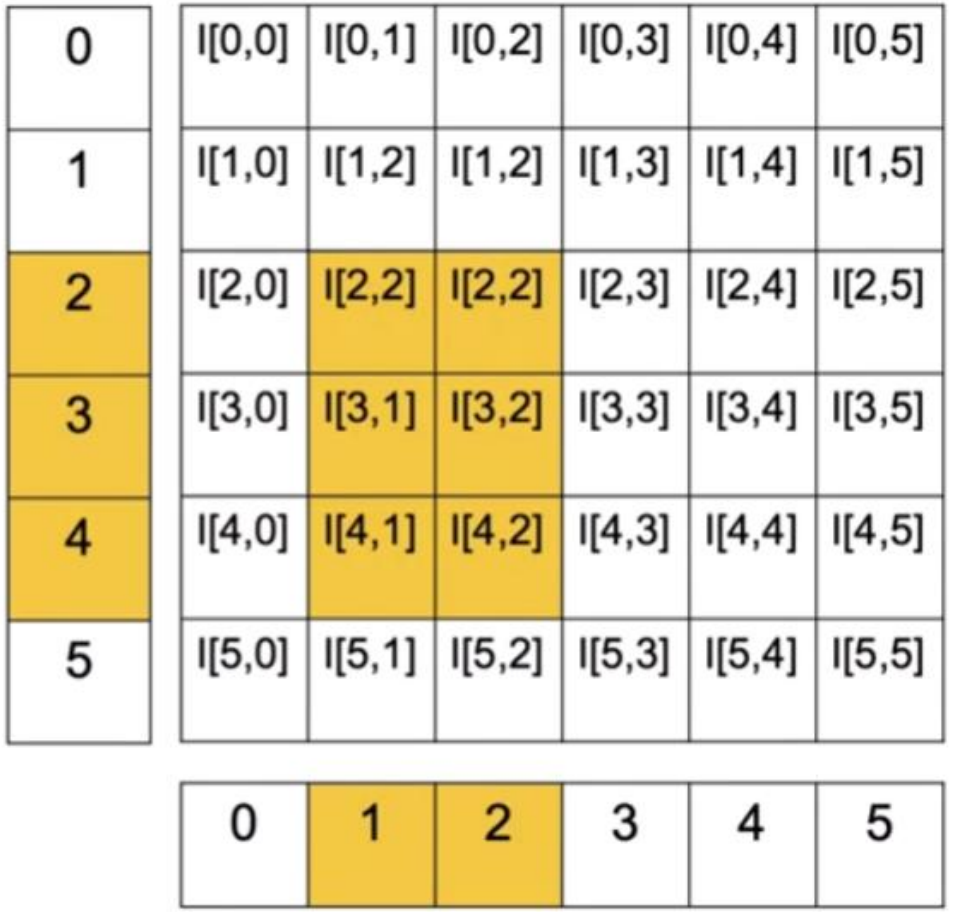
0	I[0,0]	I[0,1]	I[0,2]	I[0,3]	I[0,4]	I[0,5]
1	I[1,0]	I[1,2]	I[1,2]	I[1,3]	I[1,4]	I[1,5]
2	I[2,0]	I[2,2]	I[2,2]	I[2,3]	I[2,4]	I[2,5]
3	I[3,0]	I[3,1]	I[3,2]	I[3,3]	I[3,4]	I[3,5]
4	I[4,0]	I[4,1]	I[4,2]	I[4,3]	I[4,4]	I[4,5]
5	I[5,0]	I[5,1]	I[5,2]	I[5,3]	I[5,4]	I[5,5]

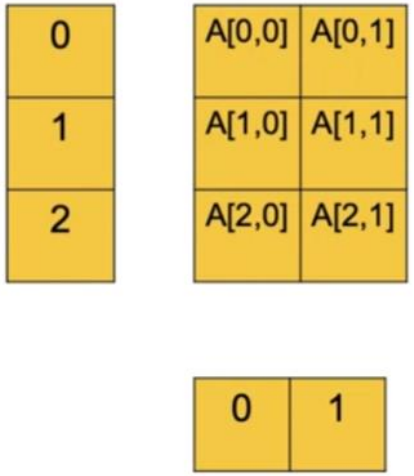
0	1	2	3	4	5

Manipulating Images - Cropping

I[2:5, 1:3]

A = I[2:5, 1:3]

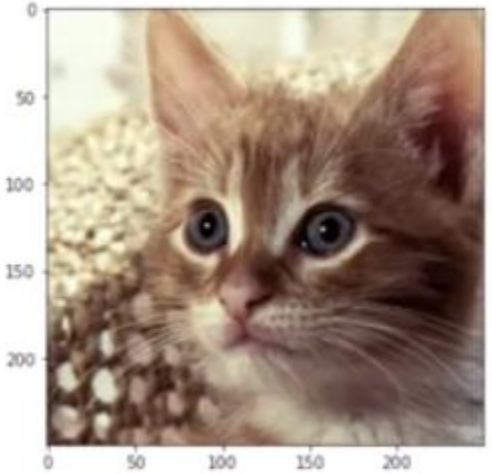




Manipulating Images

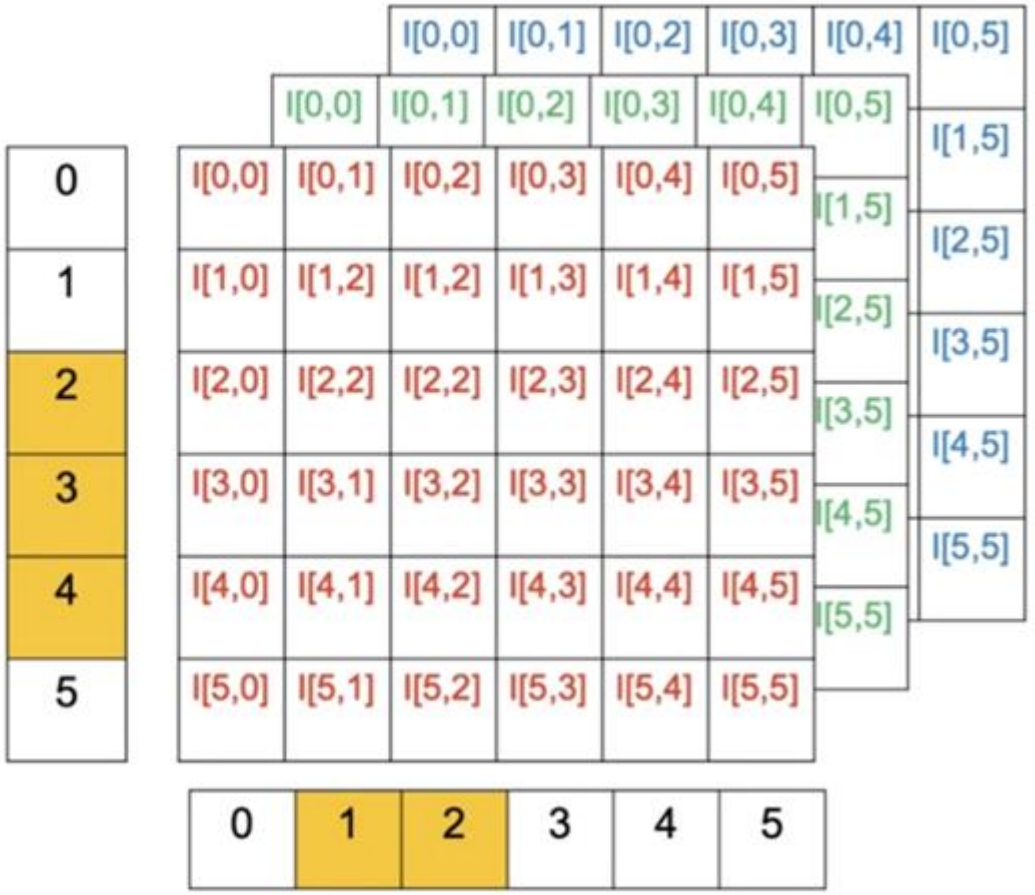
```
upper= 150
lower=400
crop_top=image[upper: lower,:,:]
```

```
left = 150
right=400
crop_horizontal=crop_top[: ,left:right,:]
```



Manipulating Images

A = I[2:5, 1:3, :]



Manipulating Images

```
from PIL import Image
import numpy as np

image = Image.open("cat.png")
image =np.array(image)
```

```
import cv2
image =cv2.imread("cat.png")
```

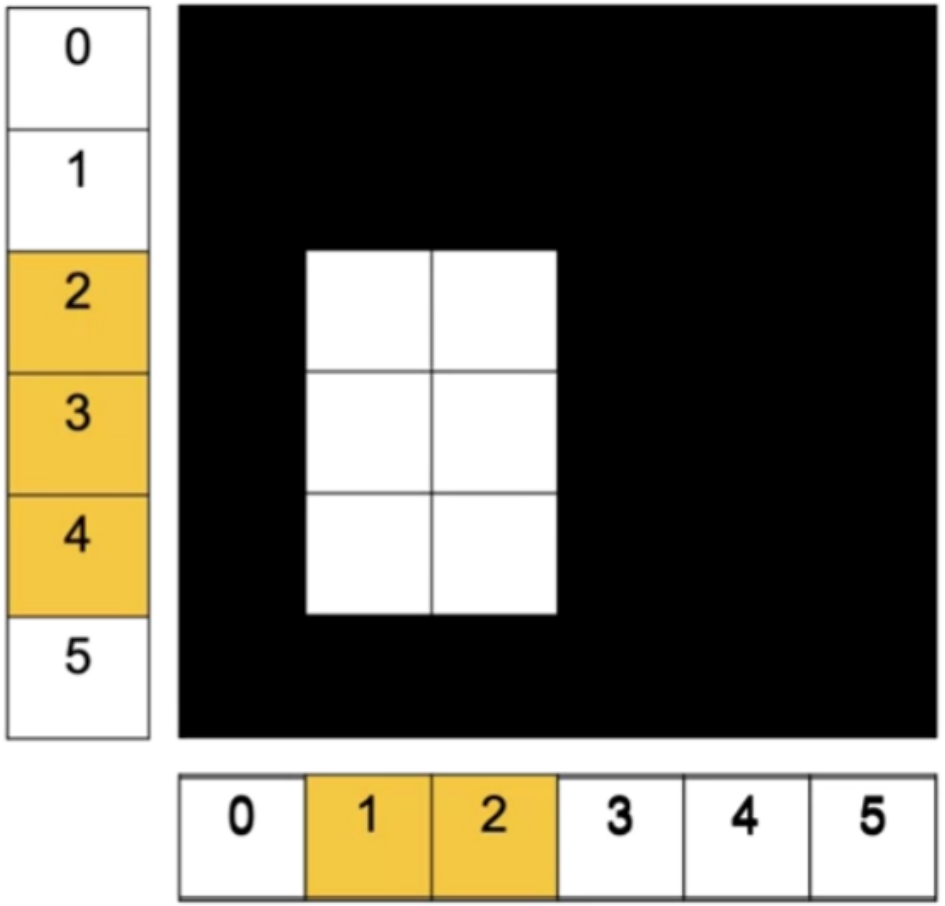
Manipulating Images

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

0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	255	255	0	0	0
3	0	255	255	0	0	0
4	0	255	255	0	0	0
5	0	0	0	0	0	0
	0	1	2	3	4	5

Manipulating Images

A = I[2:5, 1:3] = 255



Manipulating Images

I[4,1:5]=255

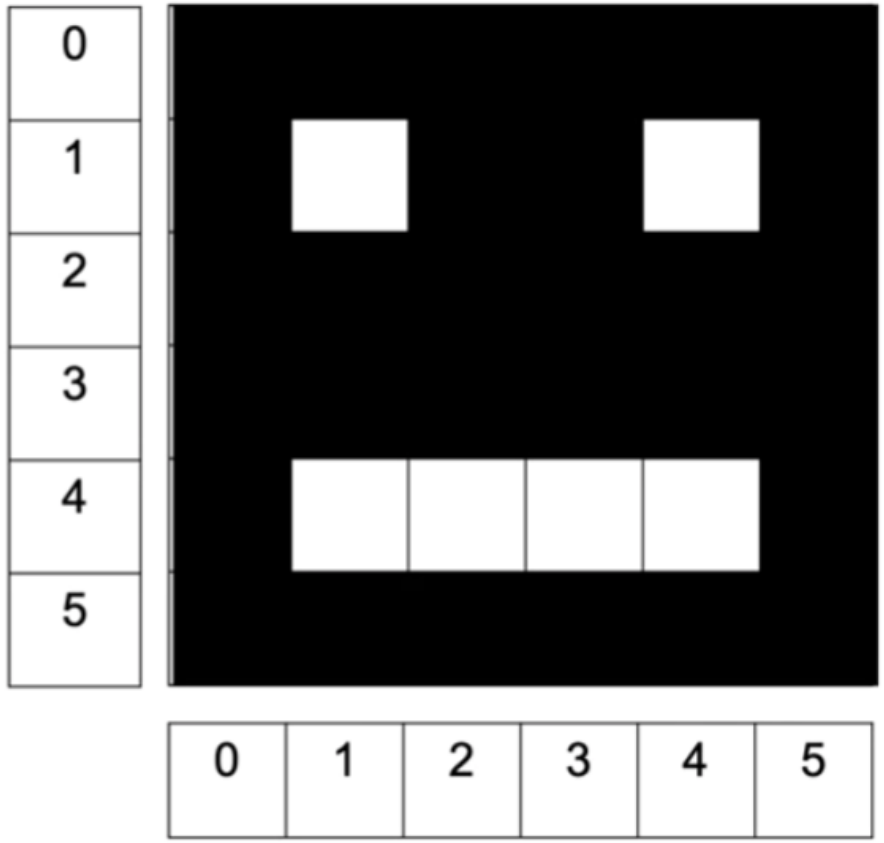
I[1,1]=255

I[1,4]=255

I:

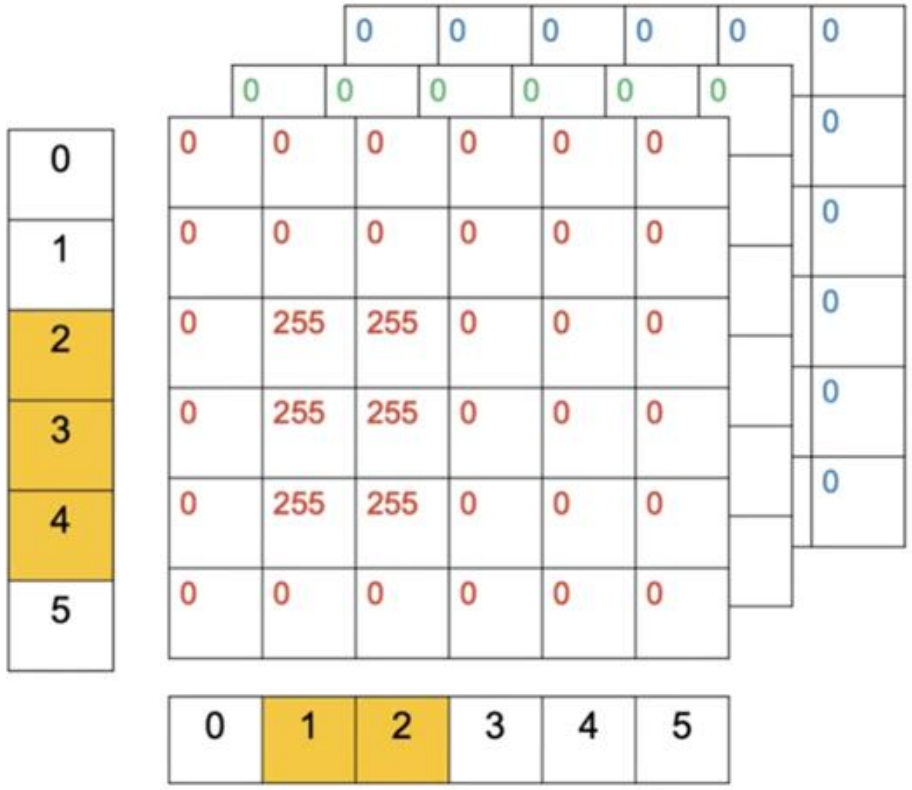
	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	255	0	0	255	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	255	255	255	255	0
5	0	0	0	0	0	0

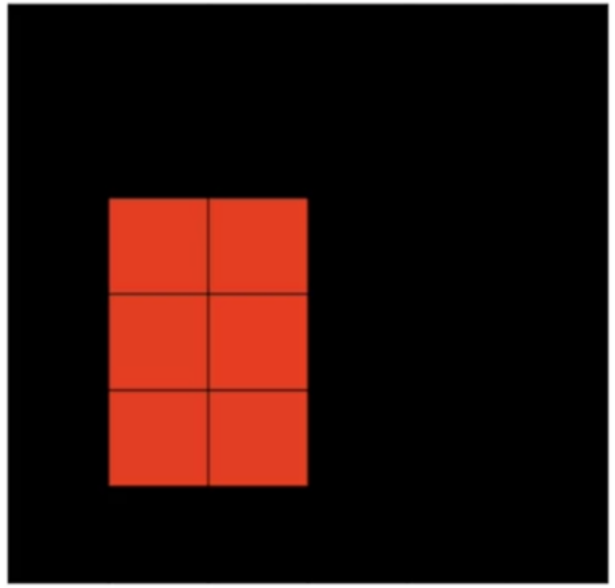
I:



Manipulating Images

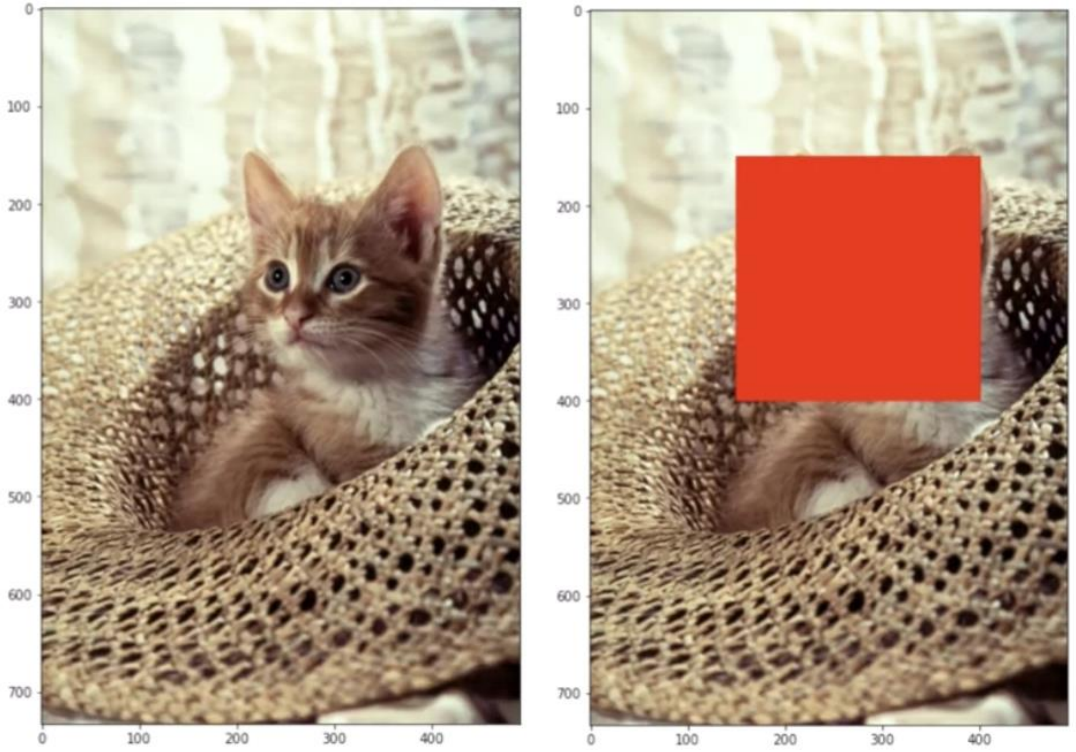
A = I[2:5, 1:3, 0] = 255





Manipulating Images

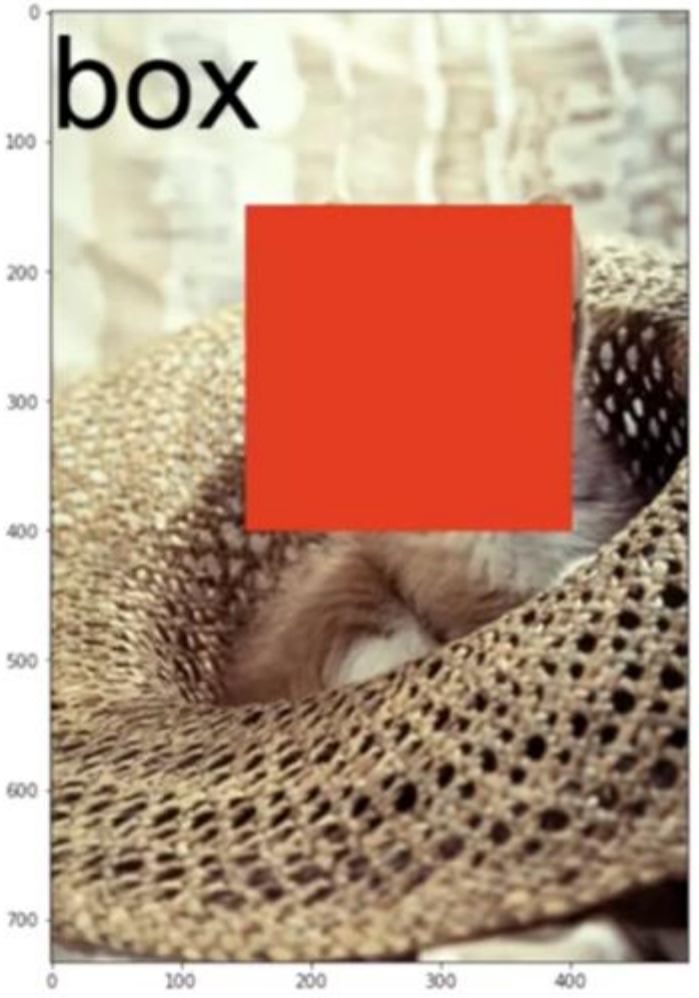
```
from PIL import  ImageDraw

image_draw=image.copy()

image_fn= ImageDraw.Draw(im=image_draw)

shape = [left, upper, right, lower]
image_fn.rectangle(xy=shape,fill="red")
```



Manipulating Images

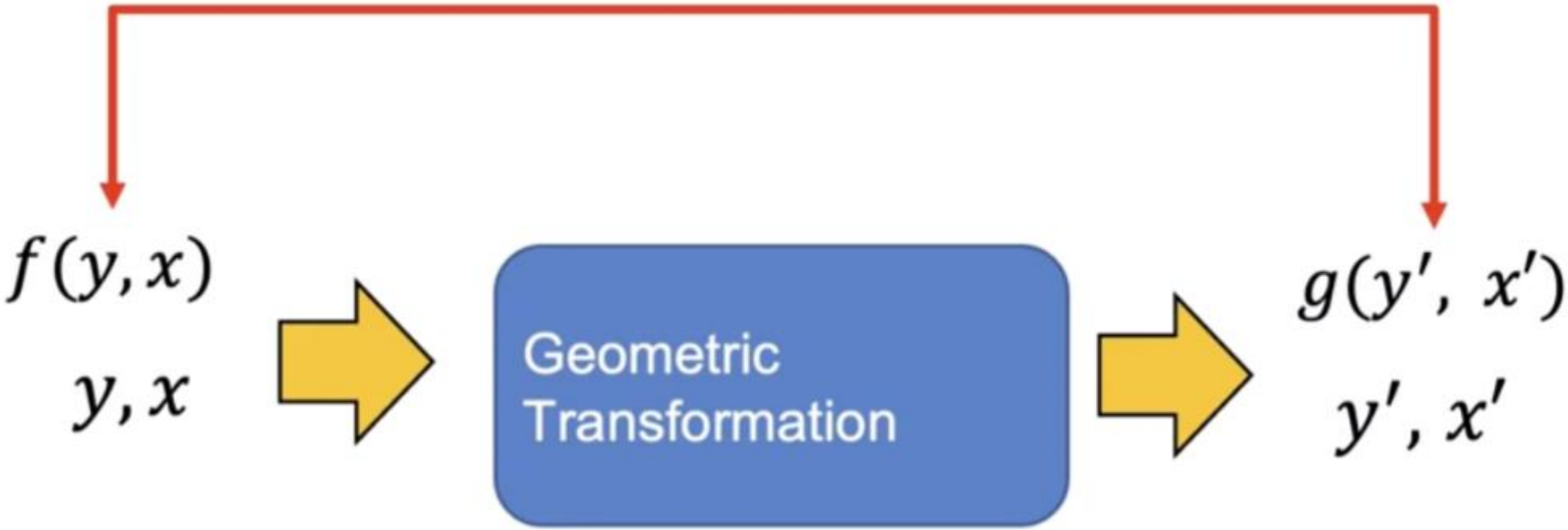
```
from PIL import ImageFont
fnt = ImageFont.truetype( '/Library/Fonts/Arial.ttf' , 100)
image_fn.text(xy=(0,0),text="box",font=fnt,fill=(0,0,0))
```

image_draw:



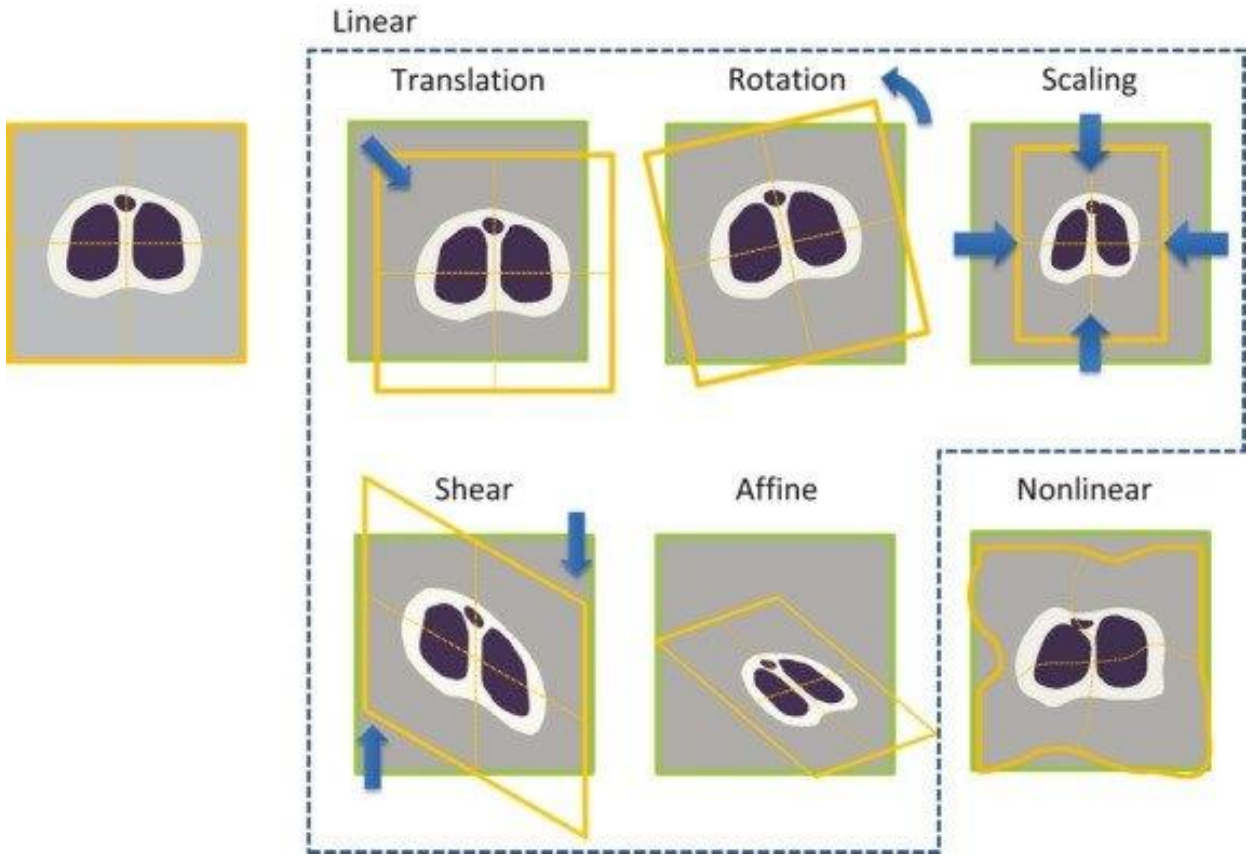
Geometric Transformation

- In a geometric transformation, we change the coordinates of the image
- Also known as Spatial Transformation

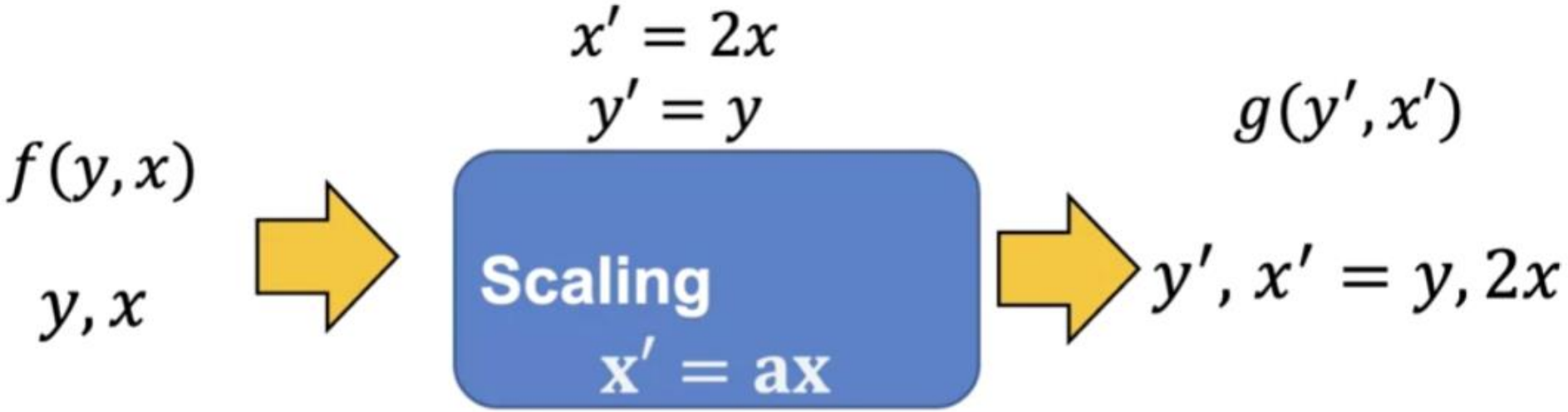


Geometric Operations

- Different types of operations can be performed to do geometric transformation of an image.



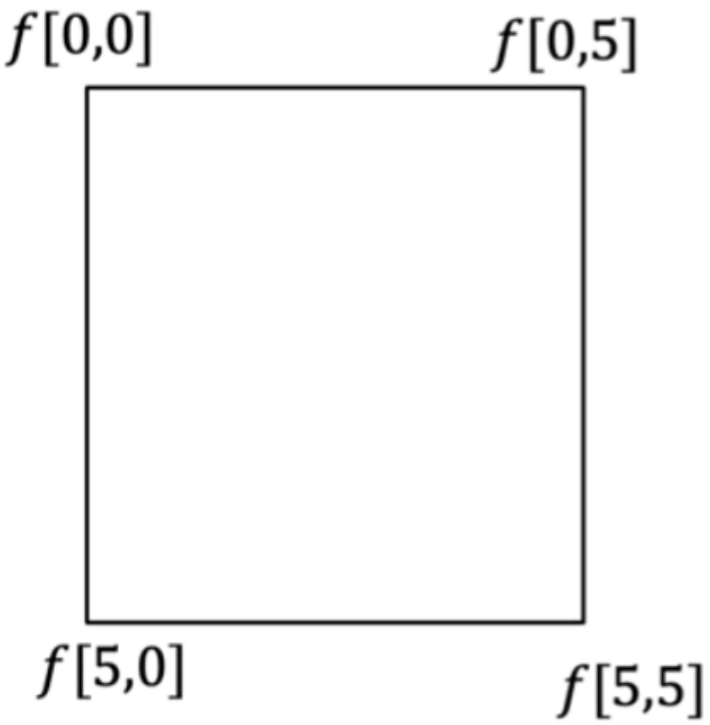
Geometric Operations - Scaling

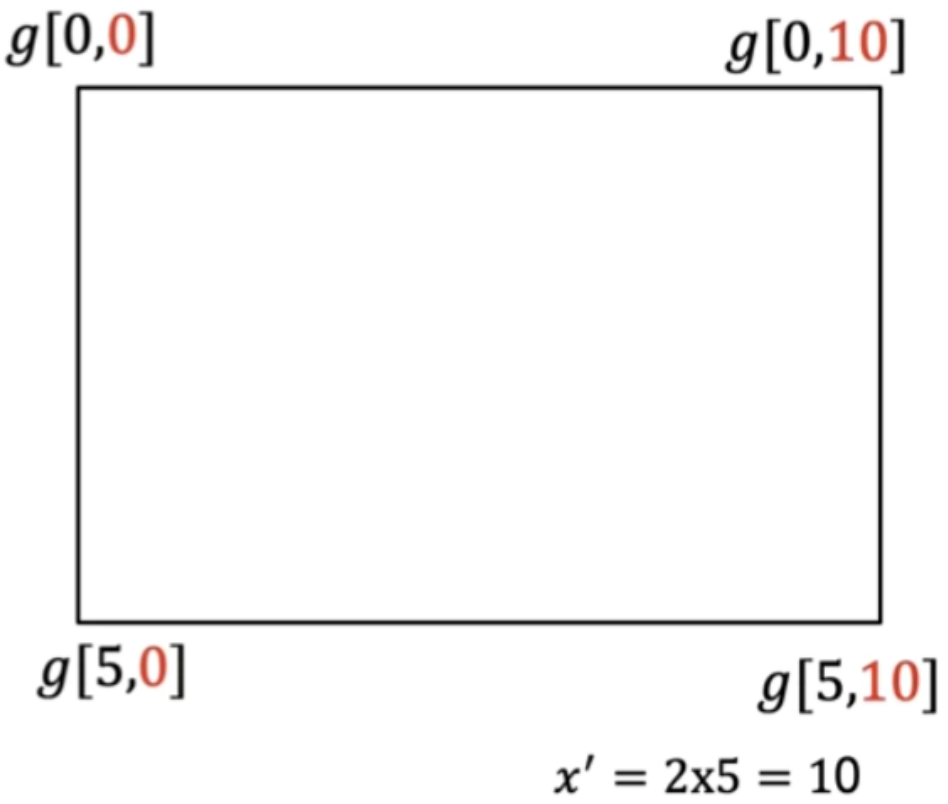


$$g(y, 2x) = f(y, x)$$

Geometric Operations - Scaling

$x' = ax$ $x' = 2x$





$x' = 2x5 = 10$

Geometric Operations - Scaling

f[0,0]	f[0,1]	f[0,2]	f[0,3]	f[0,4]	f[0,5]
f[1,0]	f[1,2]	f[1,2]	f[1,3]	f[1,4]	f[1,5]
f[2,0]	f[2,2]	f[2,2]	f[2,3]	f[2,4]	f[2,5]
f[3,0]	f[3,1]	f[3,2]	f[3,3]	f[3,4]	f[3,5]
f[4,0]	f[4,1]	f[4,2]	f[4,3]	f[4,4]	f[4,5]
f[5,0]	f[5,1]	f[5,2]	f[5,3]	f[5,4]	f[5,5]

g[0,0]	g[0,1]	g[0,2]	g[0,3]	g[0,4]	g[0,5]	g[0,6]	g[0,7]	g[0,8]	g[0,9]	g[0,10]
g[1,0]	g[1,1]	g[1,2]	g[1,3]	g[1,4]	f[1,5]	g[1,6]	g[1,7]	g[1,8]	g[1,9]	g[1,10]
g[2,0]	g[2,1]	g[2,2]	g[2,3]	g[2,4]	g[2,5]	g[2,6]	g[2,7]	g[2,8]	g[2,9]	g[2,10]
g[3,0]	g[3,1]	g[3,2]	g[3,3]	g[3,4]	g[3,5]	g[3,6]	g[3,7]	g[3,8]	g[3,9]	g[3,10]
g[4,0]	g[4,1]	g[4,2]	g[4,3]	g[4,4]	g[4,5]	g[4,6]	g[4,7]	g[4,8]	g[4,9]	g[4,10]
g[5,0]	g[5,1]	g[5,2]	g[5,3]	g[5,4]	g[5,5]	g[5,6]	g[5,7]	g[5,8]	g[5,9]	g[5,10]

Geometric Operations - Scaling

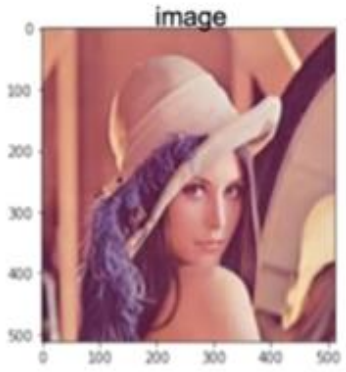
- Interpolation – to determine the value of the unknown pixels based on the neighboring pixel values

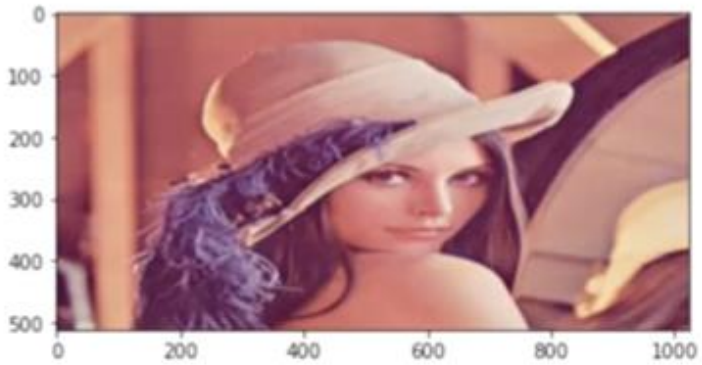
f[0,0]	f[0,1]	f[0,2]	f[0,3]	f[0,4]	f[0,5]
f[1,0]	f[1,2]	f[1,2]	f[1,3]	f[1,4]	f[1,5]
f[2,0]	f[2,2]	f[2,2]	f[2,3]	f[2,4]	f[2,5]
f[3,0]	f[3,1]	f[3,2]	f[3,3]	f[3,4]	f[3,5]
f[4,0]	f[4,1]	f[4,2]	f[4,3]	f[4,4]	f[4,5]
f[5,0]	f[5,1]	f[5,2]	f[5,3]	f[5,4]	f[5,5]

g[0,0]	g[0,0]	g[0,2]	g[0,2]	g[0,4]	g[0,4]	g[0,6]	g[0,6]	g[0,8]	g[0,8]	g[0,10]
g[1,0]	g[1,0]	g[1,2]	g[1,2]	g[1,4]	g[1,4]	g[1,6]	g[1,6]	g[1,8]	g[1,8]	g[1,10]
g[2,0]	g[2,0]	g[2,2]	g[2,2]	g[2,4]	g[2,4]	g[2,6]	g[2,6]	g[2,8]	g[2,8]	g[2,10]
g[3,0]	g[3,0]	g[3,2]	g[3,2]	g[3,4]	g[3,4]	g[3,6]	g[3,6]	g[3,8]	g[3,8]	g[3,10]
g[4,0]	g[4,0]	g[4,2]	g[4,2]	g[4,4]	g[4,4]	g[4,6]	g[4,6]	g[4,8]	g[4,8]	g[4,10]
g[5,0]	g[5,0]	g[5,2]	g[5,2]	g[5,4]	g[5,4]	g[5,6]	g[5,6]	g[5,8]	g[5,8]	g[5,10]

Geometric Operations - Scaling

```
from PIL import Image

image = Image.open("lenna.png")

width= 512
hight=512
new_width=2*width
new_hight=hight

new_image=image.resize((new_width,new_hight))
```





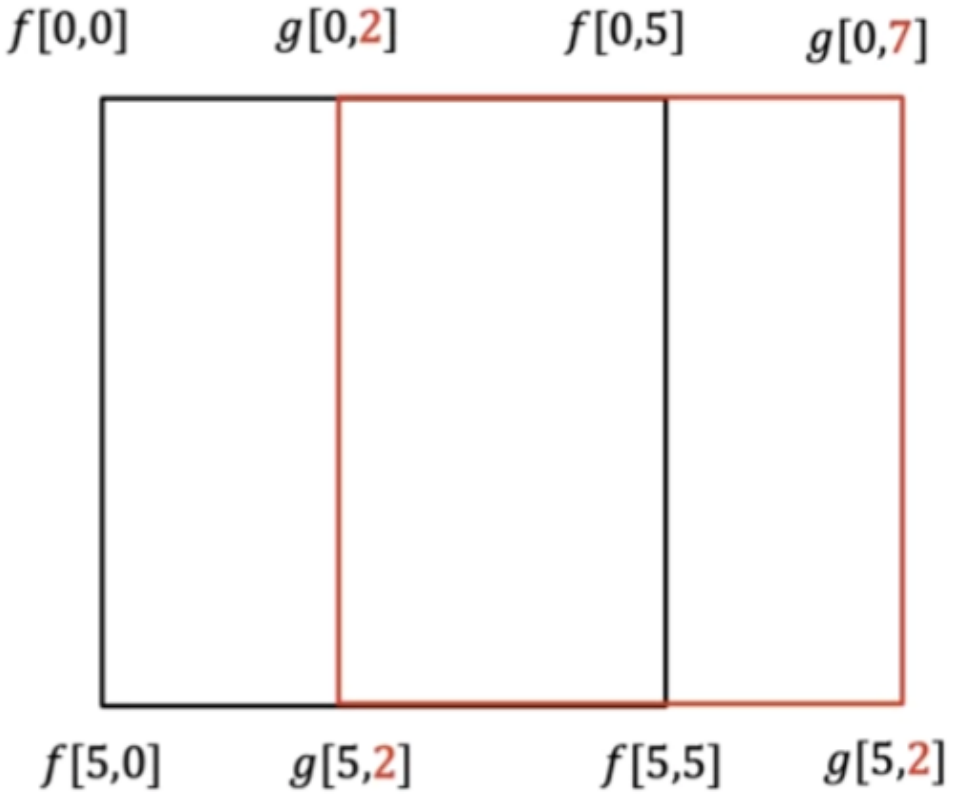
Geometric Operations - Translation

$$x' = x + t_x$$

$$x' = x + 2$$

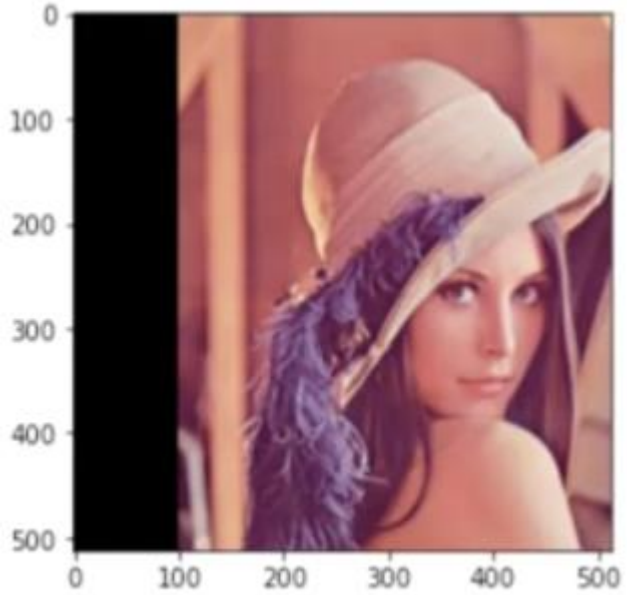
$$x' = 0 + 2 = 2$$

$$x' = 5 + 2 = 7$$

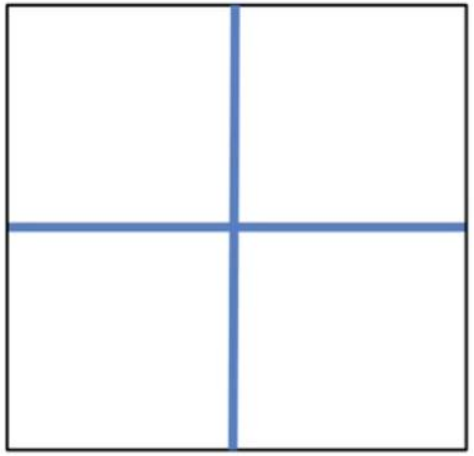


Geometric Operations - Translation

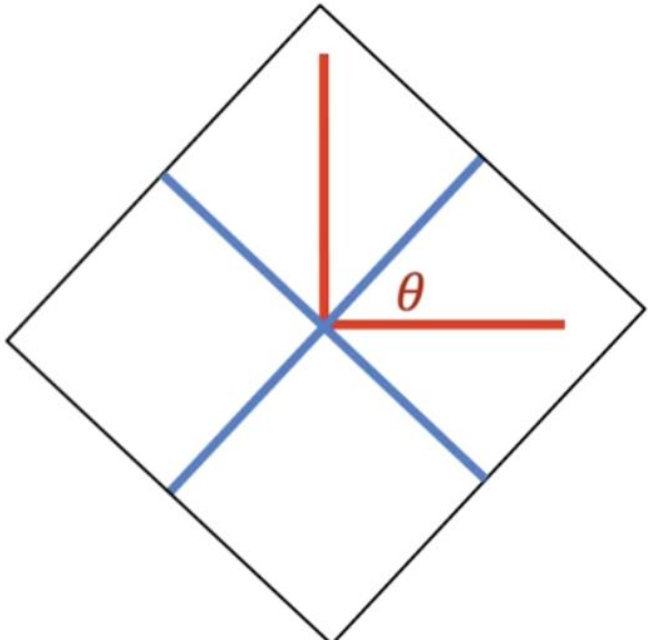
```
rows,cols,_=image.shape

tx=100
ty=0
M = np.float32([[1,0,tx],[0,1,ty]])

new_image = cv2.warpAffine(image,M,(cols,rows))
```



Geometric Operations - Rotation



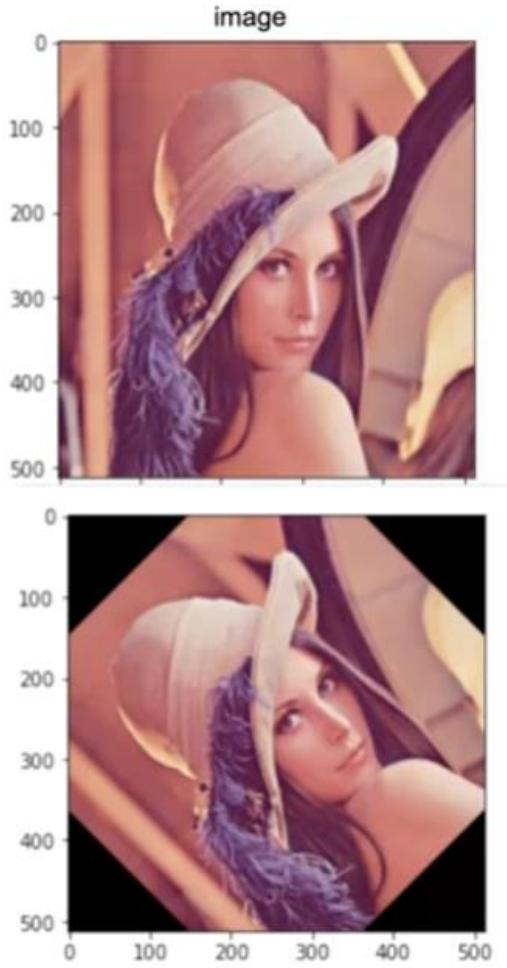
θ
Theta



$$\begin{bmatrix} r\cos(\theta) & r\sin(\theta) & stuff \\ -r\sin(\theta) & r\cos(\theta) & stuff \end{bmatrix}$$

Geometric Operations - Rotation

```
image = Image.open("lenna.png")
theta=45

new_image=image.rotate(theta)
```



IBM Developer

Some Other Geometric Operations

- Rigid Transformation – Translation + Rotation
- Similarity Transformation – Translation + Rotation + Scaling
- Affine - Translation + Rotation + Scaling + Shearing
- Achieved by changing the factors in the transformation matrix, see this video for animation: https://www.youtube.com/watch?v=AheaTd_I5Is

Geometric Transformation Matrix

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where:

- (x, y): Original coordinates of a point in the image.
- (x', y'): New coordinates after transformation.
- a, b, c, d: Elements that control rotation, scaling, and shearing.
- t_x, t_y: Elements that control translation in the x and y directions, respectively.

Geometric Transformation Matrix

Translation

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shearing

Scaling

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation

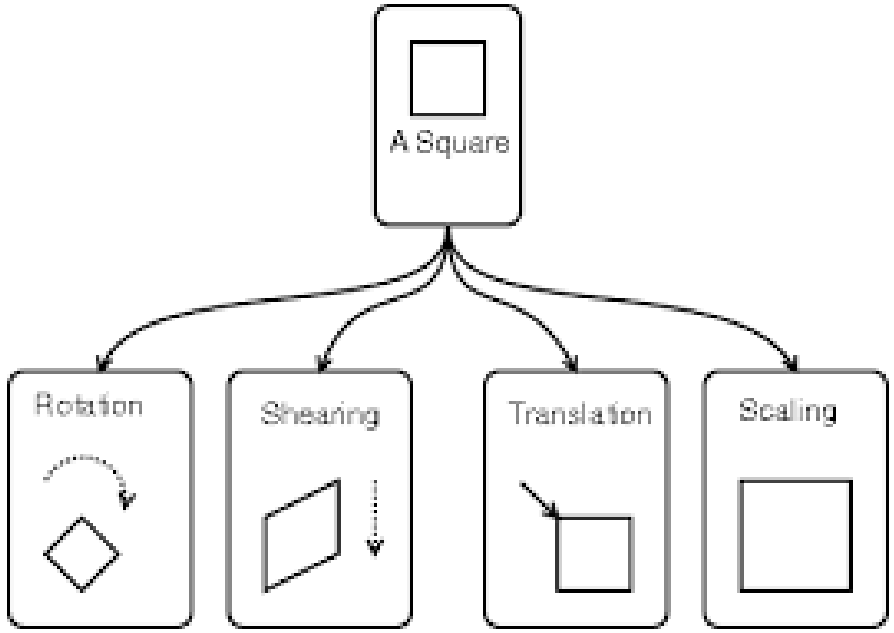
Geometric Operations - Affine

- Affine Transformations

$$x' = ax + t_x$$
$$y' = dy + t_y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

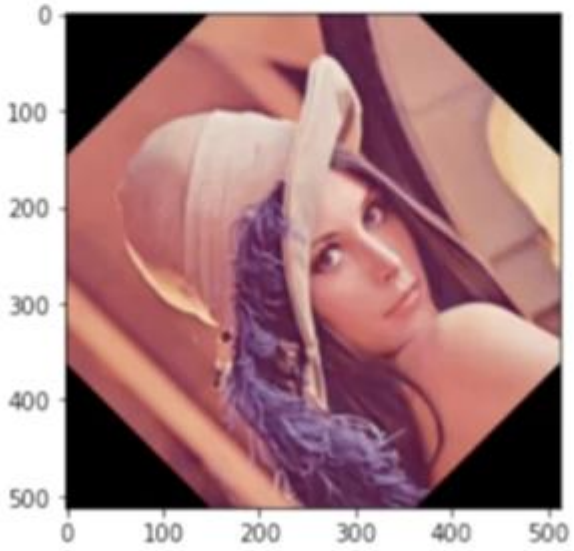
$$\begin{bmatrix} a & c & t_x \\ d & b & t_y \end{bmatrix}$$



http://www.nilab.info/docs/libart/libart-affine-transformation-matrices.html

Geometric Operations - Affine

```
theta=45.0
M = cv2.getRotationMatrix2D(center=(cols//2-1,rows//2-1),angle=theta,scale=1)

new_image = cv2.warpAffine(image,M,(cols,rows))
```



Geometric Transformation Matrix - Affine

Combining rotation, scaling, shearing, and translation gives the general affine transformation:

$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

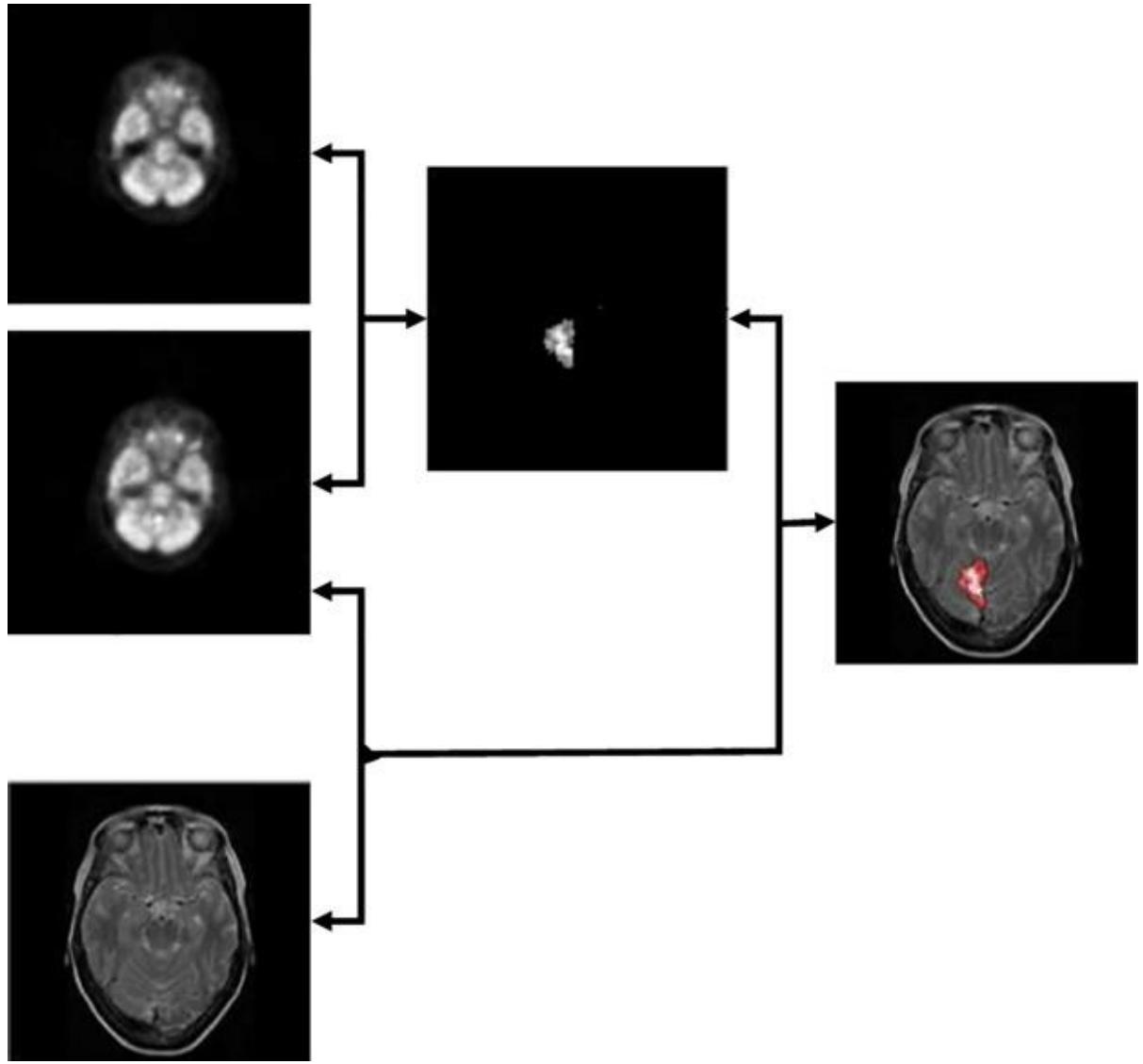
where the elements a, b, c, and d are calculated based on the specific transformations desired:

- $a = s_x \cdot \cos(\theta) - sh_y \cdot \sin(\theta)$
- $b = s_x \cdot \sin(\theta) + sh_y \cdot \cos(\theta)$
- $c = -sh_x \cdot \cos(\theta) + s_y \cdot \sin(\theta)$
- $d = sh_x \cdot \sin(\theta) + s_y \cdot \cos(\theta)$

Why Geometrics Transformation?

- Alignment of images taken at different time stamps, from different angles and from different devices for better understanding of the domain.

- Example: Brain Image Registration taken from different devices CT, XRAY, MRI etc



Acknowledgements

Some of the content of these slides is taken from:

- www.coursera.com
- Digital Image Processing by Gonzalez