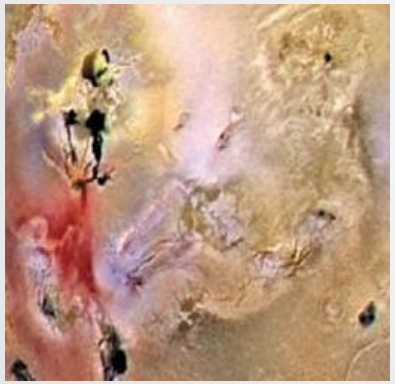






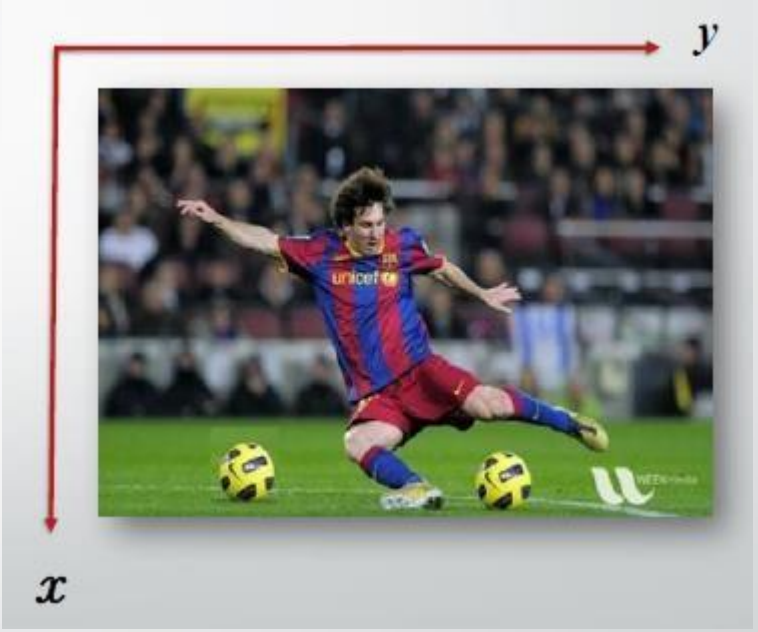

CSC 3141

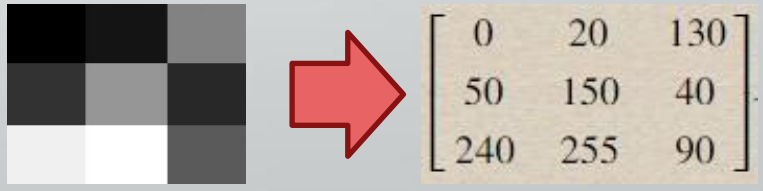
IMAGE PROCESSING LABORATORY

03 – Image Processing with OpenCV

What is an image?

- An image is a 2-Dlight intensity function
  f(x,y)
- An image is considered as a matrix
- A digital image f(x,y) is described both in spatial coordinates and Brightness
- The points in the image and element value of matrix identifies gray level value at that point This element is called a Pixel.

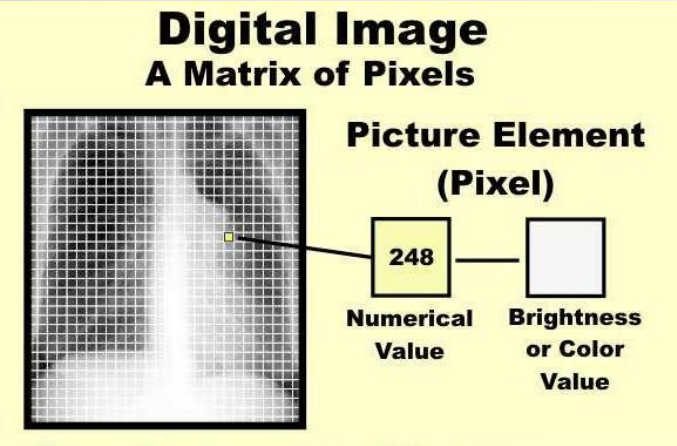




Matrix or digital representation of an image

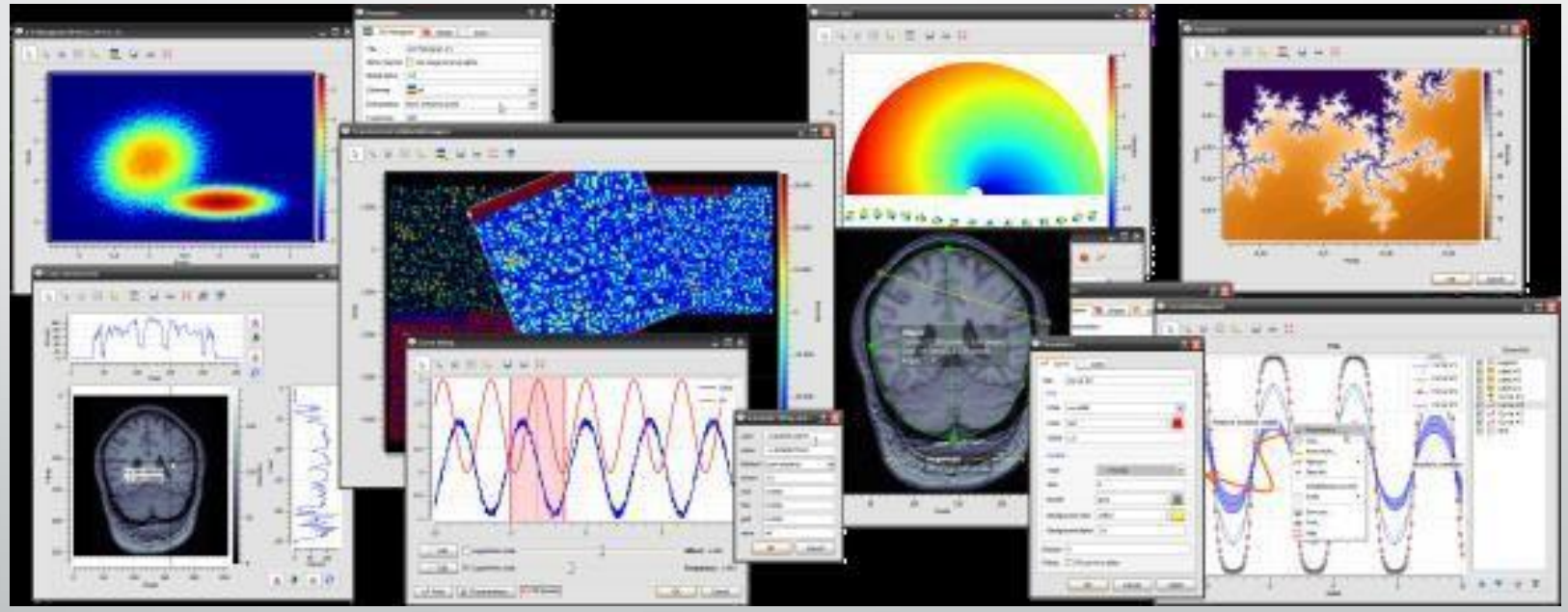
- A digital image is a matrix of many small elements, or pixels that store intensity values.
- Each pixel is represented by a numerical value.

$$I = \begin{pmatrix} f(1,1) & f(1,2) & \cdots & f(1,M \\ f(2,1) & f(2,2) & \cdots & f(2,M \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ f(N,1) & f(N,2) & \cdots & f(N,M \end{pmatrix}$$



What is image processing ?

- Analysis and manipulation of a digitized image, especially in order to improve its quality.



Types of Digital Images

Binary Images

Each pixel is just Black or White ie. 0 or 1.

Ex. image shape : (280 x 450)

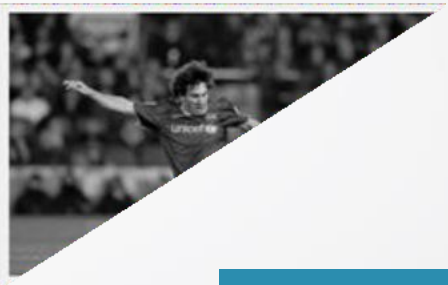
1	1	1	1	1
0	1	1	1	1
0	0	0	1	1
0	0	0	0	0
0	0	0	0	0

Binary

Gray-scale images

Each pixel is shade of Gray, from black (0) to white (255), ie. each pixel ~8 bits

Ex. image shape : (280 x 450)



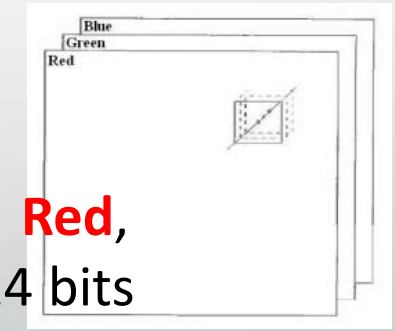
138	145	149	150	149
117	135	146	147	143
80	100	119	134	141
55	64	80	101	116
50	48	52	63	75

Gray-scale

Color or RGB (BGR) Images

Each pixel corresponds to 3 color bands **Red**, **Green** and **Blue** values,), ie. each pixel ~24 bits

Ex. image shape : (280 x 450 x 3)



Color / RGB

```
imgB =

   71   91  110
   50   57   73
   46   44   47


imgG =

   79   98  117
   55   63   79
   51   48   52


imgR =

   86  107  126
   58   68   84
   50   49   53
```

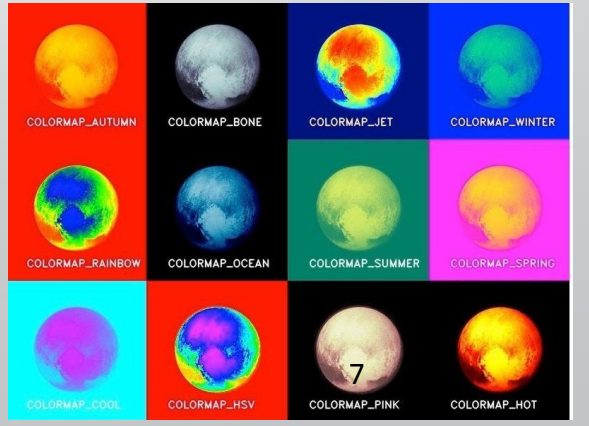
Data types of images

Data Type	Description	Range
int8	8 bit integer	[-128 , 127]
uint8	8 bit unsigned integer	[0, 255]
int16	16 bit integer	[-32768 , 32767]
uint16	16 bit unsigned integer	[0 , 65535]
int32	32 bit integer	$[-(2^{31} - 1) , 2^{31} - 1]$
uint32	32 bit unsigned integer	$[0 , 2^{32} - 1]$
double	Double precision real number	Machine Specific

Color Maps

- A mapping from 0-255 values to 256 colors (When considering an 8-bit image)
- COLORMAP_*. Lower grayscale values are replaced by colors to the left of the scale while higher grayscale values are to the right of the scale
- There are 12 different color maps in OpenCV
- Useful for improved visualization of **grayscale image** small intensity variations

Class	Scale
COLORMAP_AUTUMN	
COLORMAP_BONE	
COLORMAP_COOL	
COLORMAP_HOT	
COLORMAP_HSV	
COLORMAP_JET	
COLORMAP_OCEAN	
COLORMAP_PINK	
COLORMAP_RAINBOW	
COLORMAP_SPRING	
COLORMAP_SUMMER	
COLORMAP_WINTER	



Using OpenCV-Python

OpenCV-Python

- **OpenCV** is a huge C/C++ library of programming functions mainly aimed at real-time computer vision and digital image processing. **OpenCV-Python** is a wrapper package for OpenCV python bindings

- **Installation**

  Within the terminal :

```
pip install opencv-python
pip install opencv-contrib-python
```

- **Importing OpenCV**

```
import cv2
```

Read an Image

Syntax:

```
img_array = cv2.imread( image_name , image_mode )
```

image_name: image name if it is in working directory or full path of the image otherwise

image_mode: is a flag which specifies the way image should be

- cv2.IMREAD_COLOR or 1: Loads an image in RGB mode
- cv2.IMREAD_GARYSCALE or 0: Loads an image in grayscale mode
- cv2.IMREAD_UNCHANGED or -1: Loads an image with alpha channel if applicable
- cv2.IMREAD_ANYCOLOR or 4: Loads image in any possible color format

```
import cv2

img = cv2.imread('roi.jpg',cv2.IMREAD_GRAYSCALE)
#or
img = cv2.imread('roi.jpg',0)
```

Display an image using OpenCV

Syntax:

```
cv2.imshow( window_name , image_array )
```

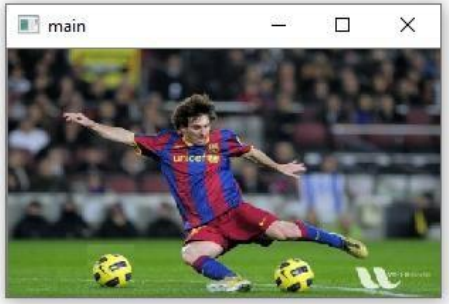
window_name : a name for the cv2 window
image_array **:** opened image as an array-like object

- Additional functions required

  `cv2.waitKey()` : allows users to display a window for given milliseconds or until any key is pressed

  `cv2.destroyAllWindows()`: Destroys all the windows created

```
import cv2
import numpy as np

img = cv2.imread('roi.jpg',cv2.IMREAD_COLOR)
cv2.imshow('main',img)
#cv2.waitKey()
cv2.waitKey(5000)
cv2.destroyAllWindows()
```



Display multiple images using OpenCV

- **Using Horizontal Stacking**

```
import cv2
import numpy as np

img = cv2.imread('roi.jpg',cv2.IMREAD_COLOR)

cv2.namedWindow("main", cv2.WINDOW_NORMAL)
cv2.imshow('main', np.hstack((img,img,img)))

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

- **Using Vertical Stacking**

```
import cv2
import numpy as np

img = cv2.imread('roi.jpg',cv2.IMREAD_COLOR)

cv2.namedWindow("main", cv2.WINDOW_NORMAL)
cv2.imshow('main', np.vstack((img,img,img)))

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

Display an image using Matplotlib

Syntax:

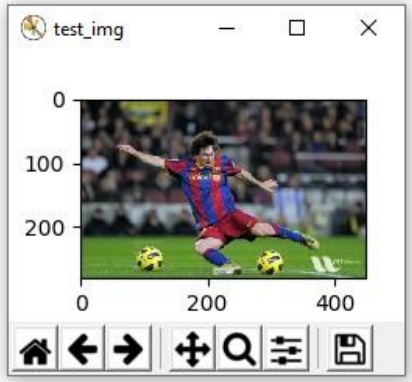
```
plt.imshow(image_array)
```

image_array : opened image as an array-like object

- Additional functions required

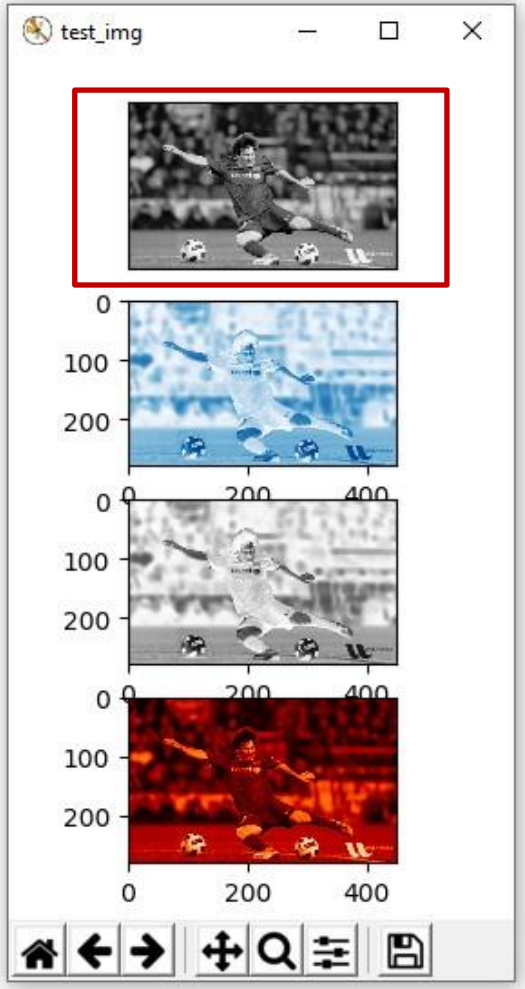
  `cv2.cvtColor(img_arr, color_conversion)`: to convert cv2 BGR array to RGB
  `plt.figure()` : to change manipulate figure properties
  `plt.show()` : to display the pyplot figure

```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('roi.jpg',cv2.IMREAD_COLOR)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

plt.figure(figsize =(2, 2),num='test_img') #width,height in inches
plt.imshow(img)
plt.show()
```



Display multiple images using Matplotlib

```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('roi.jpg',0)

plt.figure(figsize =(6, 6),num='test_img')

plt.subplot(4,1,1)
plt.imshow(img,cmap='Greys_r')
plt.xticks([]),plt.yticks([])

plt.subplot(412)
plt.imshow(img,cmap='Blues')

plt.subplot(413)
plt.imshow(img,cmap='Greys')

plt.subplot(414)
plt.imshow(img,cmap='gist_heat')

plt.show()
```



https://matplotlib.org/stable/tutorials/colors/colormaps.html

Write/save an image

- **Using OpenCV**

  Syntax:

  ```
  cv2.imwrite( filename , image_array )
  ```

  filename : a filename with acceptable image extension [.jpeg, .jpg, .png, .tiff, …]
  image_array : opened image as an array-like object

  ```
  cv2.imwrite('test_.jpg',img)
  ```

- **Using Matplotlib**

  Syntax:

  ```
  plt.savefig(filename)
  ```

  filename : a filename with acceptable image extension [.jpg, .png, .tiff, .pdf,…]

  ```
  plt.savefig('test.jpg')
  ```

Image basic properties

- Accessing pixel values `img[:,2,1]  # row, col, channel`
- Shape `img.shape  # row, col, channel`
- Check pixel data type `img.dtype`
- Check image type `type(img)`
- Convert pixel data type `img.astype('uint8')  | np.uint8(img)`

Changing Color space

Syntax:

```
img_array = cv2.cvtColor( input_image, conversion_type)
```

input_image : opened image as an array-like object

conversion_type : a flag which specifies the way image should be converted

- BGR to Gray : cv2.COLOR_BGR2GRAY
- BGR to HSV : cv2.COLOR_BGR2HSV
- BGR to RGB : cv2.COLOR_BGR2RGB
- BGR to RGBA: cv2.COLOR_BGR2RGBA

https://docs.opencv.org/4.x/d8/d01/group__imgproc__color__conversions.html

Change Color Map

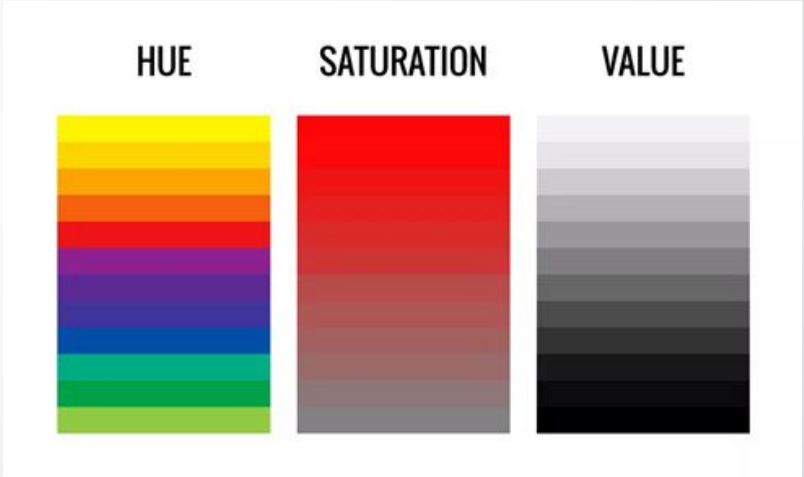
```python
import numpy as np
import cv2

img = cv2.imread('roi.jpg',1)

im1 = cv2.applyColorMap(img,cv2.COLORMAP_AUTUMN)
# or
im1 = cv2.applyColorMap(img, 7)
```

0	COLORMAP_AUTUMN	
1	COLORMAP_BONE	
2	COLORMAP_JET	
3	COLORMAP_WINTER	
4	COLORMAP_RAINBOW	
5	COLORMAP_OCEAN	
6	COLORMAP_SUMMER	
7	COLORMAP_SPRING	
8	COLORMAP_COOL	
9	COLORMAP_HSV	
10	COLORMAP_PINK	
11	COLORMAP_HOT	

https://docs.opencv.org/4.x/d3/d50/group__imgproc__colormap.html

OpenCV Color Model – HSV/HSB

- The Hue in HSV represents the color, Saturation in HSV represents the greyness, and Value in HSV represents the brightness.

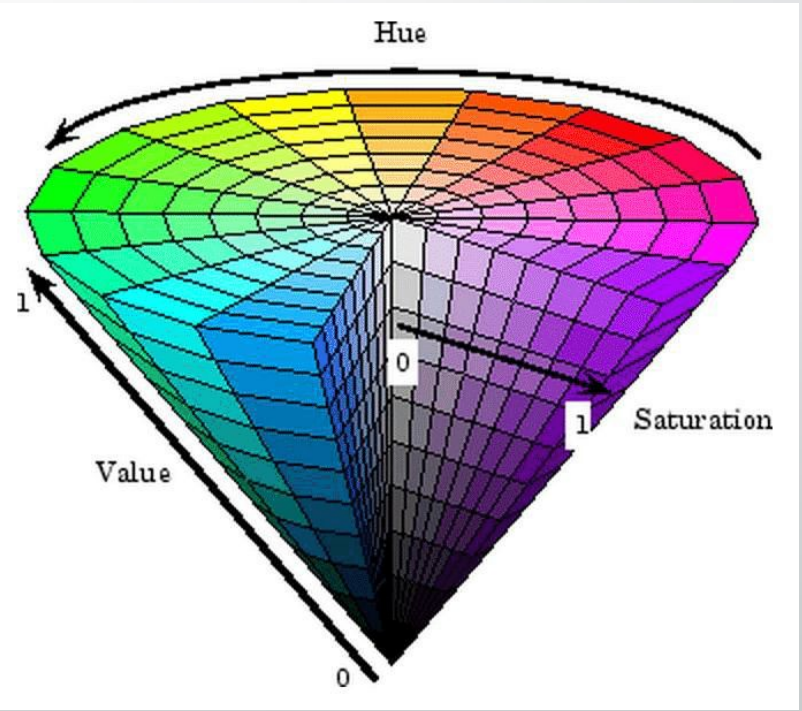


- **Hue :** Hue is the color portion of the color model [0,179] . 2^0 = 1 hue value
- **Saturation :** The saturation (S) of a color describes how white the color is [0,255]
- **Value :** The value refers to the lightness or darkness of a color [0,255].

Find related HSV values for RGB values

```
import numpy as np
import cv2

green = np.uint8([[[0,255,0 ]]])
red   = np.uint8([[[0,0,255 ]]])#BGR
blue  = np.uint8([[[255,0,0 ]]])

hsv_green = cv2.cvtColor(green,cv2.COLOR_BGR2HSV)
hsv_red   = cv2.cvtColor(red,cv2.COLOR_BGR2HSV)
hsv_blue  = cv2.cvtColor(blue,cv2.COLOR_BGR2HSV)

print( hsv_green )
print( hsv_red )
print( hsv_blue )
```

```
[[[ 60 255 255]]]
[[[  0 255 255]]]
[[[120 255 255]]]
```



Color Regions Detection

Real Time Demo

https://discuss.dizzycoding.com/how-to-find-the-red-color-regions-using-opencv/

https://docs.opencv.org/4.x/df/d9d/tutorial_py_colorspaces.html

https://en.wikipedia.org/wiki/Web_colors#HTML_color_names

– END –