

# Introduction to Digital Images

In Python

# Digital image representation

An image is defined as a two-dimensional function,  $F(x,y)$ , where  $x$  and  $y$  are spatial coordinates, and the amplitude of  $F$  at any pair of coordinates  $(x,y)$  is called the **intensity** of that image at that point, or **pixel**.



$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & f(0,2) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & f(1,2) & \dots & f(1,N-1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & f(M-1,2) & \dots & f(M-1,N-1) \end{bmatrix}$$

200	205	...	...
210	206	...	...
...	...	...	...
208	210	...	...

Each pixel is a measure of brightness (intensity of light)

The size of image is specified as **width \* height** (number of pixels)

# Image types

**Binary images** are images for which the pixels have only two possible intensity values. These are displayed as black and white, where each pixel of the image has the value 0 or 1 representing black and white.



**grayscale images**, the result is a two-dimensional array with the number of rows and columns equal to the number of pixel rows and columns in the image. Low numeric values indicate darker shades and higher values lighter shades. The range of pixel values is often 0 to 255. We divide by 255 to get a range of 0 to 1.



**(RGB) Color images** are represented as three-dimensional Numpy arrays - a collection of three two-dimensional arrays, one each for red, green, and blue channels. Each one, like grayscale arrays, has one value per pixel and their ranges are identical.



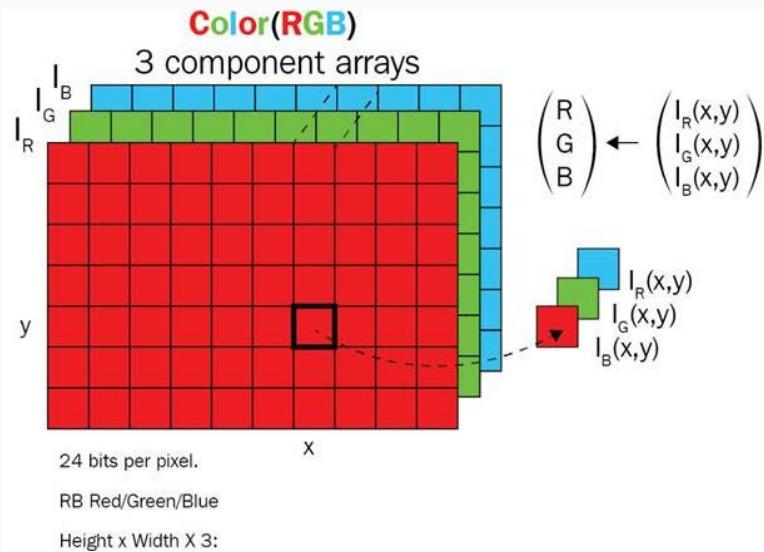
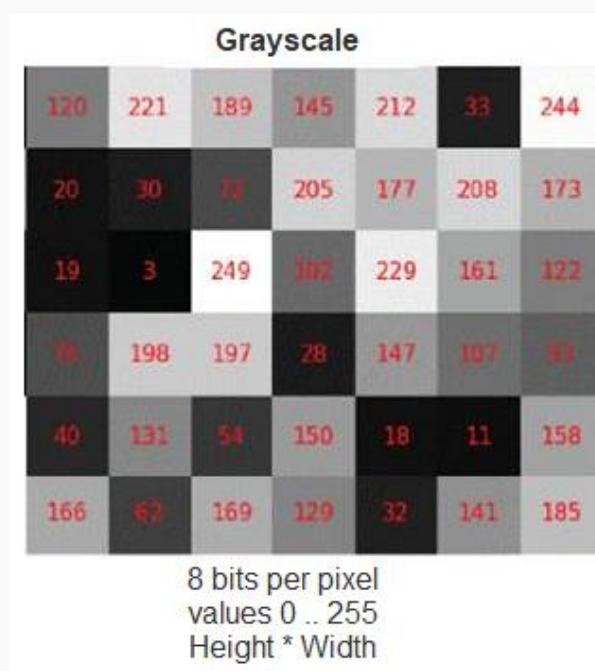
# Images as multidimensional arrays

**Single channel** → Binary images , grayscale images

**Multi channel** → Color images

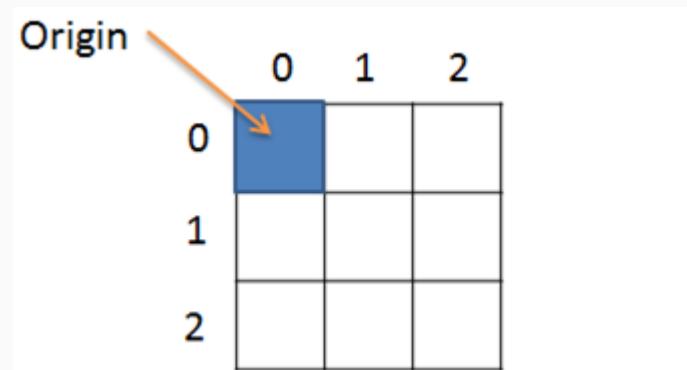
Black & white							
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0

1 bit per pixel  
values 0 / 1  
Height \* Width



## Read / Write image libraries

- python provides a lot of libraries to process images and do the corresponding tasks such as read, write, and display images. For example imageio, OpenCV (Open Source Computer Vision), Matplotlib, PIL (Python Imaging Library), Pillow.



## Reading an image,

- When you think of digital images, you probably think of sampled image formats such as the JPEG,GIF, .....
- To **read / load** an image from a specific file use the following code :

```
import imageio.v3 as iio
# read an image
img = iio.imread("C:\\\\Users\\\\user\\\\Desktop\\\\aa.jpeg")
```

- To **get properties** of an image (number of rows, columns and channels, type of image data, number of pixels etc.):

```
print(type(img))
print('shape: ', img.shape)
print(type(img.shape))
print('size: ', img.size)
print('ndim: ', img.ndim)
print('dtype: ', img.dtype)
```

## Image properties

`shape` returns a tuple of integers representing the size of the ndarray in each dimension, To assign each value to a variable, unpack the tuple as follows:

```
h, w, c = img.shape  
print('width: ', w)  
print('height: ', h)  
print('channels:', c)
```

Of course, you can also directly access them by index.

```
print('width: ', img.shape[1])  
print('height:', img.shape[0])
```

# Writing/Saving an Image

```
# write an image on D disk in png format  
io.imwrite(uri: "D:\\lina.png", img)
```

The first parameter is a string representing the **file name**. The filename must include:

- image path (*optional*)
- image **name**
- image **format** like .jpg, .png, etc.

The second parameter is the **variable** that stores the image that we want to write (save).

# Display an Image

Python has many different libraries that can be used to display images. We will use matplotlib.

`plt.imshow()` draws an image on the current figure (creating a figure if there isn't a current figure).

`plt.show()` displays the figure. You shouldn't call it until you've plotted things and want to see them displayed.

```
import matplotlib.pyplot as plt  
  
# display an image  
plt.title('My image')  
plt.imshow(img)  
plt.show()
```

# Pixel representation

Pixels are the smallest unit of an image which consists of four components

**Red, Green, Blue** and in short (**RGB**).

The value of all the components lie between 0 and 255 both inclusive.

Zero means the component is absent and 255 means the component is fully present.

Since,  $2^8 = 256$ . So, 8 bits can represent any value in the range 0 to 255. This means we will need only **8 bits to store the value of any of the three component**.

And since a pixel has 3 components so, the total number of bits required to store the value of all the three components RGB is equal to  $3 \times 8 = \text{24 bits or 3 bytes}$ .

# Get and set Pixels

To **get** value of pixel with index (100,100) we write:

```
# get pixel values R+G+B  
px = img[100, 100];  
# get Red value of pixel  
red = img[100, 100, 0];
```

To **modify** value of pixel with index (100,100) and make it white we write:

```
# modify pixel value  
img[100, 100] = [255, 255, 255];
```

Another pixel accessing and editing method :

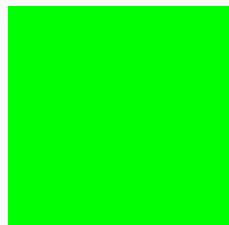
```
# print Blue value  
print('red: ', img.item(10, 10, 2))  
  
# modifying Blue value  
img.itemset((10, 10, 2), 100)
```

# Creating a random Green image

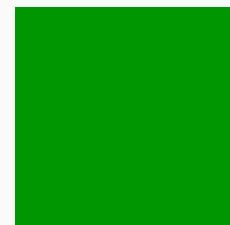
```
imarray = numpy.zeros((100, 100, 3))
imarray[:, :, 1] = 255
```

```
img = imarray.astype(numpy.uint8)
io.imwrite( url: "D:\\\\Green.jpg", img)
```

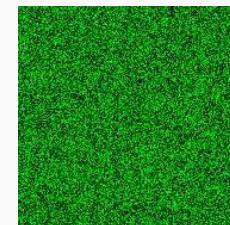
1. Modify the code to make it dark green image
2. Modify the code to generate random green image



Original



Dark



Random

# Splitting Image Channels

The B,G,R channels of an image can be split into their individual planes when needed. Then, the individual channels can be merged back together to form a BGR image again.



```
img = iio.imread("C:\\Users\\user\\Desktop\\mat.png")
red = img[:, :, 0];
green = img[:, :, 1];
blue = img[:, :, 2];
```

# Splitting Image Channels

What should we do to visualize the image in red color like this ?



# Homework

Suppose you have the left image which contains a ball, write python code to copy this ball to another place on the ground.

