OpenCV Python Tutorial: Computer Vision With OpenCV In Python

Alok Yadav

What Is Computer Vision?

Computer Vision is an interdisciplinary field that deals with how computers can be made to gain a high-level understanding from digital images or videos.

The idea here is to automate tasks that the human visual systems can do. So, a computer should be able to recognize objects such as that of a face of a human being or a lamppost or even a statue.

How Does A Computer Read An Image?

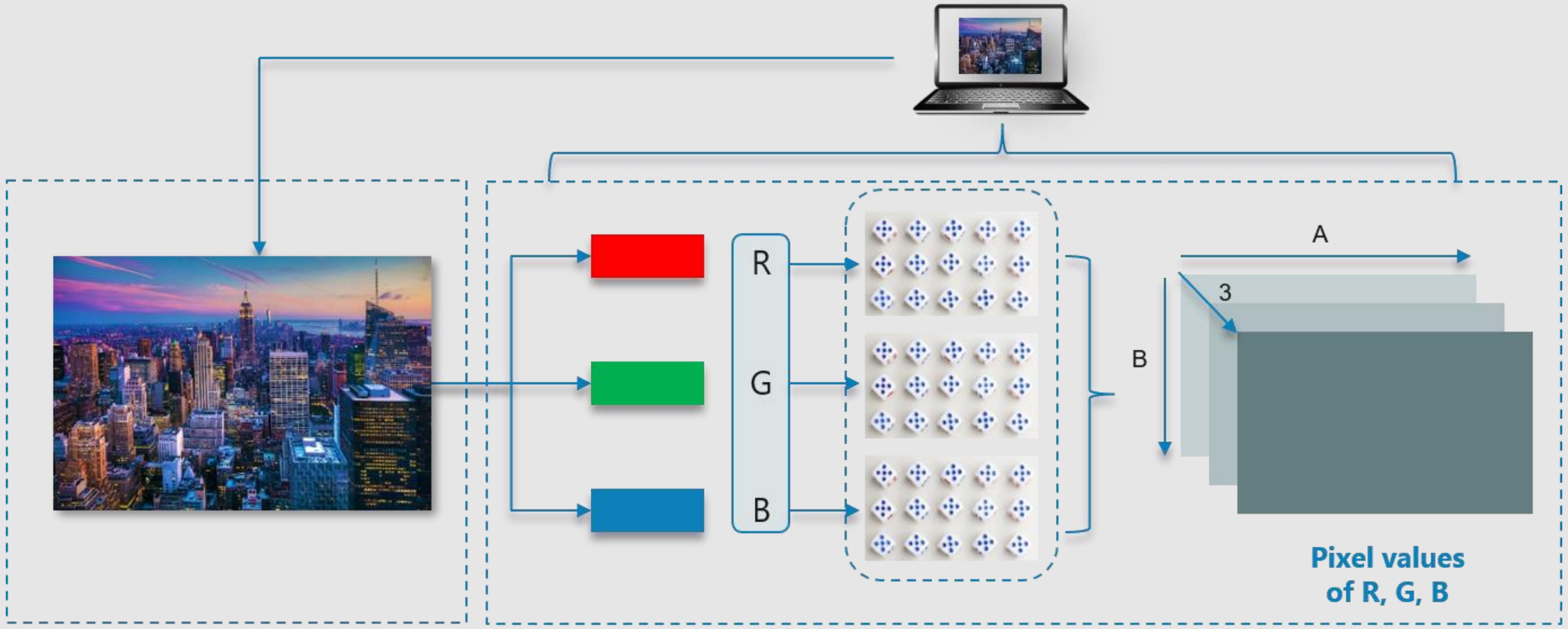
The computer reads any image as a range of values between 0 and 255.

For any color image, there are 3 primary channels – Red, green and blue. How it works is pretty simple.

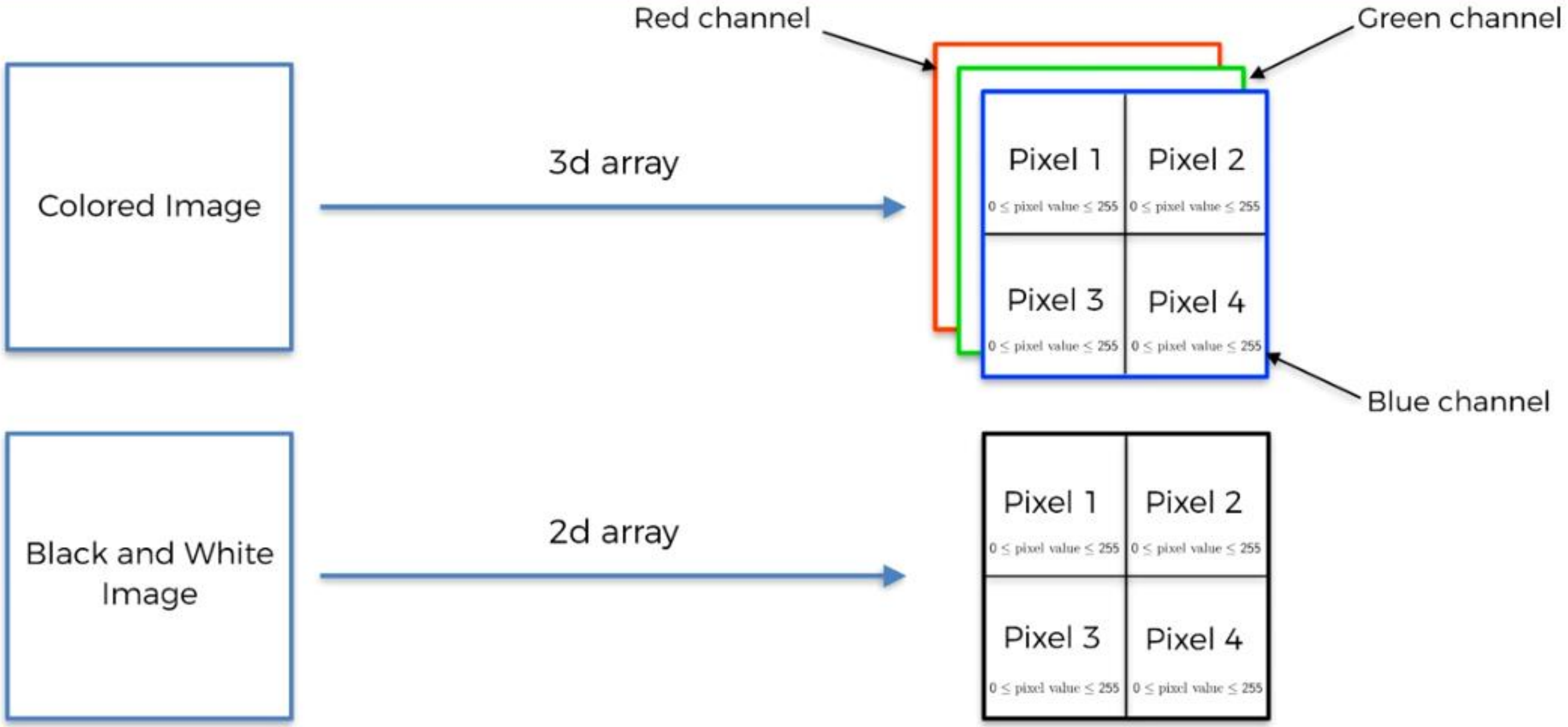
A matrix is formed for every primary color and later these matrices combine to provide a Pixel value for the individual R, G, B colors.

Each element of the matrices provide data pertaining to the intensity of brightness of the pixel.

- As shown, the size of the image here can be calculated as B x A x 3.
- Note: For a black-white image, there is only one single channel.



Input Images are converted into arrays


Red channel
Green channel
Colored Image
3d array
Pixel 1
$0 \leq$ pixel value ≤ 255
Pixel 2
$0 \leq$ pixel value ≤ 255
Pixel 3
$0 \leq$ pixel value ≤ 255
Pixel 4
$0 \leq$ pixel value ≤ 255
Blue channel
Black and White Image
2d array
Pixel 1
$0 \leq$ pixel value ≤ 255
Pixel 2
$0 \leq$ pixel value ≤ 255
Pixel 3
$0 \leq$ pixel value ≤ 255
Pixel 4
$0 \leq$ pixel value ≤ 255

What Is OpenCV?

OpenCV is a Python library which is designed to solve computer vision problems. OpenCV was originally developed in 1999 by Intel but later it was supported by Willow Garage.

OpenCV supports a wide variety of programming languages such as C++, Python, Java etc. Support for multiple platforms including Windows, Linux, and MacOS.

OpenCV Python is nothing but a wrapper class for the original C++ library to be used with Python. Using this, all of the OpenCV array structures gets converted to/from NumPy arrays.

This makes it easier to integrate it with other libraries which use NumPy. For example, libraries such as SciPy and Matplotlib.

Basic Operations With OpenCV?

- **Loading an image using OpenCV:**
- we can read the image using **imread** module.
- The 1 in the parameters denotes that it is a color image. If the parameter was 0 instead of 1, it would mean that the image being imported is a black and white image.
- **Image Shape/Resolution:**
- **Displaying the image:**
- We use the i**mshow function** to display the image by opening a window. There are 2 parameters to the imshow function which is the name of the window and the image object to be displayed.
- Later, we wait for a user event. waitKey makes the window static until the user presses a key. The parameter passed to it is the time in milliseconds.
- And lastly, we use destroyAllWindows to close the window based on the waitForKey parameter.

Resizing the image:

Resize function is used to resize an image to the desired shape. The parameter here is the shape of the new resized image.

Later, do note that the image object changes from img to resized_image, because of the image object changes now.

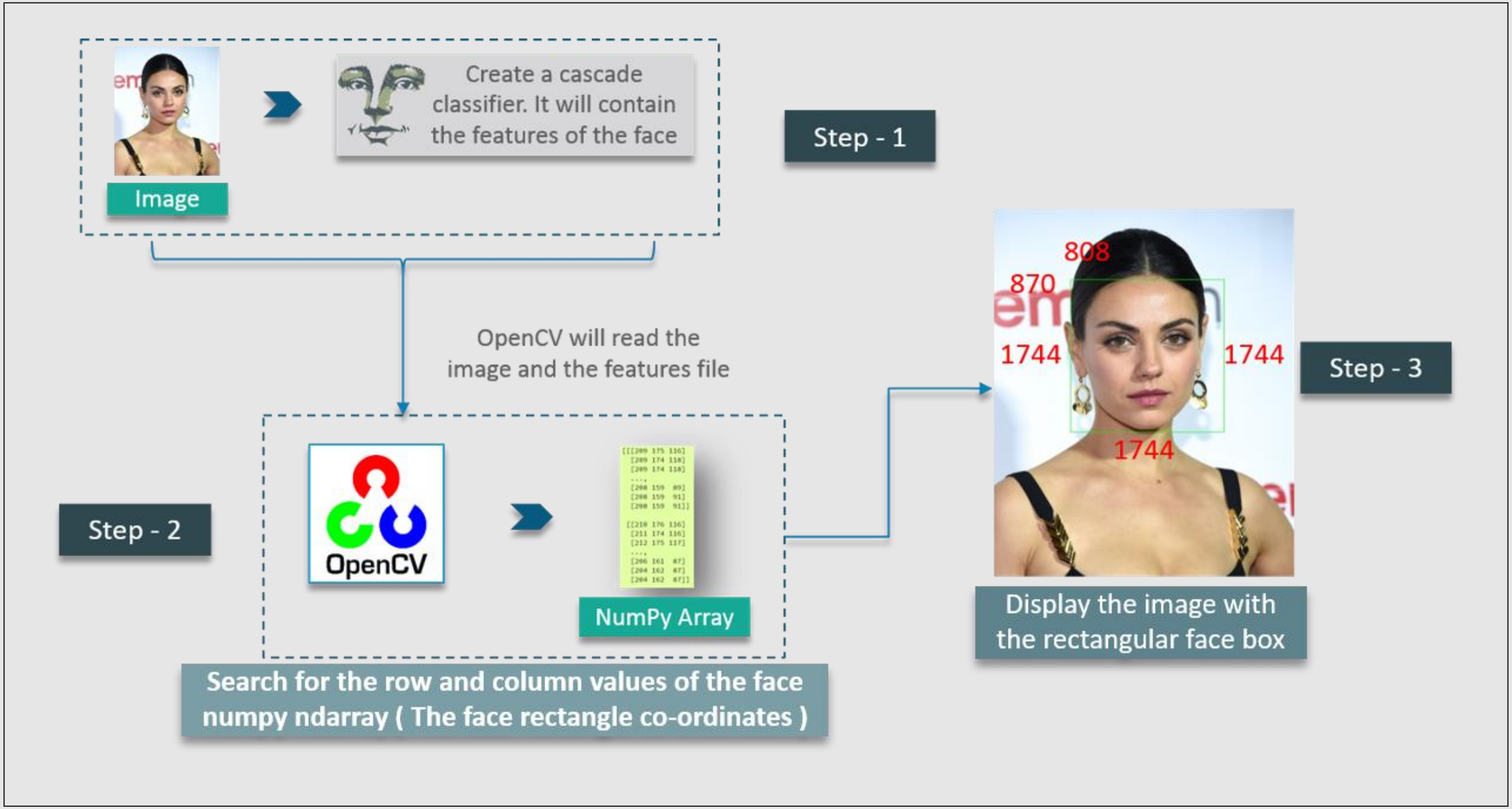
Face Detection Using OpenCV

Step 1: Considering our prerequisites, we will require an image, to begin with. Later we need to create a cascade classifier which will eventually give us the features of the face.

Step 2: This step involves making use of OpenCV which will read the image and the features file. So at this point, there are NumPy arrays at the primary data points.

All we need to do is to search for the row and column values of the face NumPy ndarray. This is the array with the face rectangle coordinates.

Step 3: This final step involves displaying the image with the rectangular face box.



Create a cascade classifier. It will contain the features of the face

OpenCV will read the image and the features file

Search for the row and column values of the face numpy ndarray (The face rectangle co-ordinates)

Display the image with the rectangular face box

First, we create a CascadeClassifier object to extract the features of the face as explained earlier.

The path to the XML file which contains the face features is the parameter here.

The next step would be to read an image with a face on it and convert it into a black and white image using COLOR_BGR2GREY.

Followed by this, we search for the coordinates for the image.

This is done using detectMultiScale.

What coordinates, you ask? It’s the coordinates for the face rectangle.

The scaleFactor is used to decrease the shape value by 5% until the face is found.

So, on the whole – Smaller the value, greater is the accuracy.