Core operations

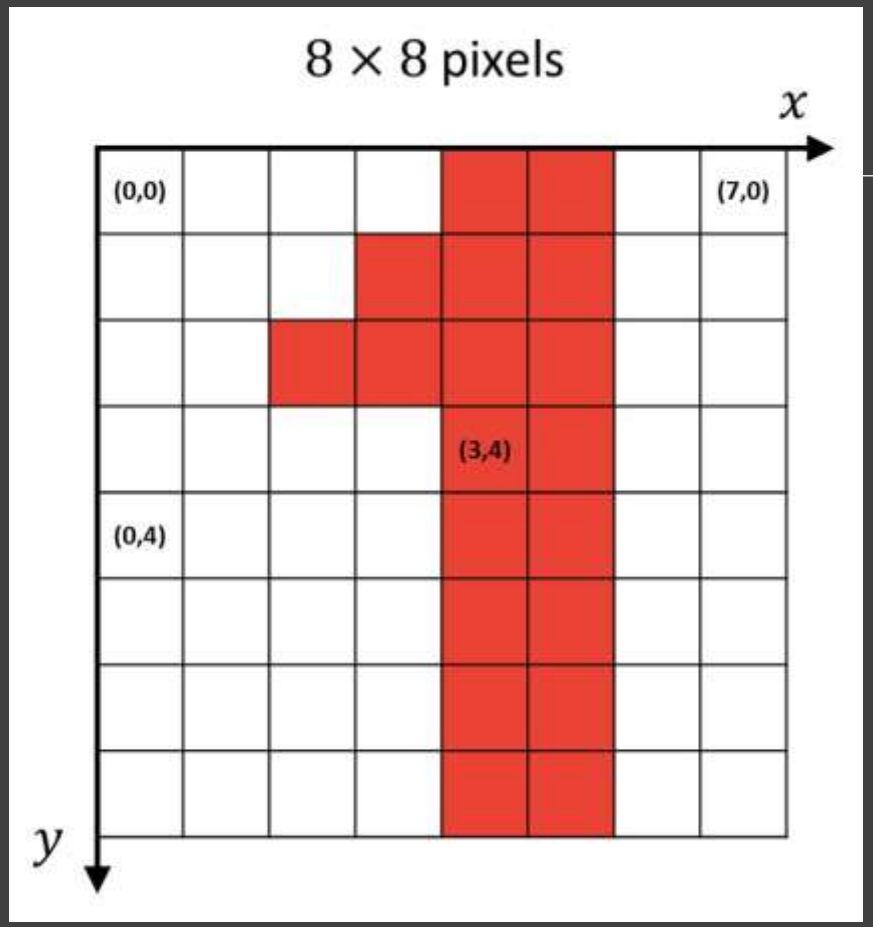
- Basic operations
- Accessing pixel in image using OpenCv
- Splitting and merging image

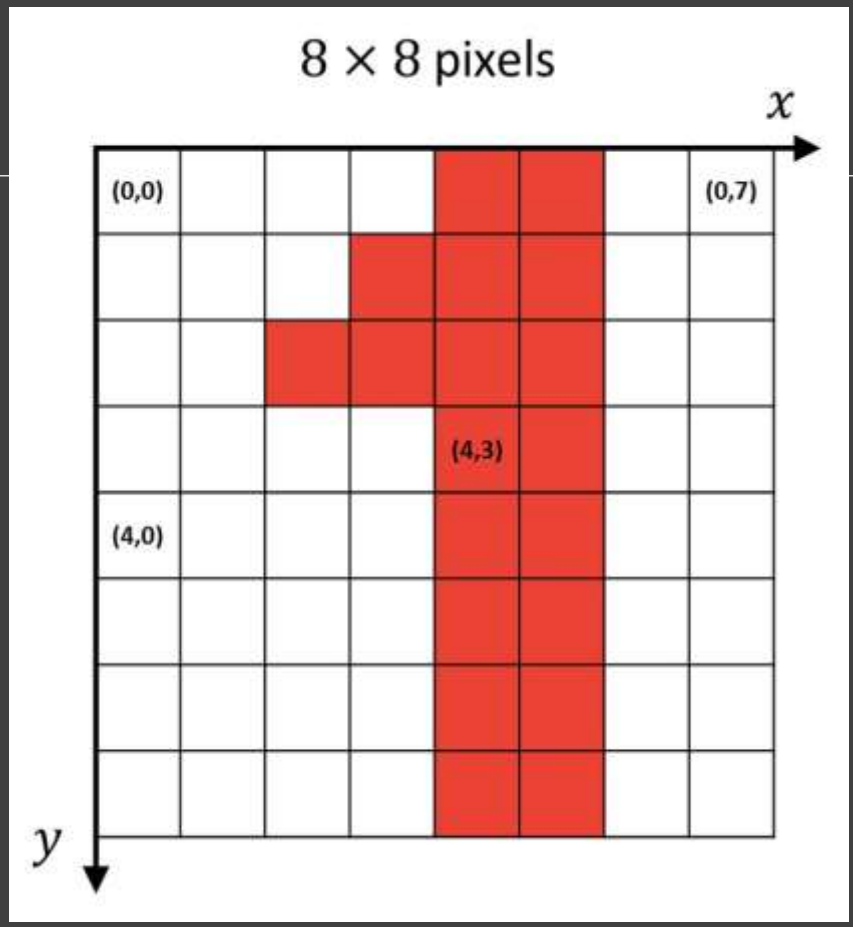
What is a pixel?

SMALLEST ELEMENT OF AN AN IMAGE

EVERY IMAGE CONSIST OF A SET OF PIXEL

Note that OpenCV loads the color images in reverse order so that the blue channel is the first one, the green channel is the second, and the red channel is the third (BGR).


8 × 8 pixels
x
y
(0,0)
(7,0)
(3,4)
(0,4)


8 × 8 pixels
x
y
(0,0)
(0,7)
(4,3)
(4,0)

Accessing of pixel

First we need to load the color image fow which we will use *cv2.imread()*

We can access a pixel value by its row and column co-ordinates

To display image we will use *cv2.imshow()*

For BGR image , it returns an array of blue ,green and red values

For gray scale image corresponding intensity i.e (different shades of gray between black and white)

Ex : px = img[100,100]

print(px)

Accessing Image properties

Image properties include number of rows, columns and channels, type of image data, number of pixels etc.

Img.shape : Shape of image is accessed by img.shape. It returns a tuple of number of rows, columns and channels (if image is color).If image is grayscale, tuple returned contains only number of rows and columns

Img.size : Total number of pixels is accessed by img.size

Img.dtype : Image datatype is obtained by img.dtype

It is very important while debugging because a large number of errors in OpenCV-Python code is caused by invalid datatype

Padding

Making **borders** images

If we want to create border around the image use

cv2.copyMakeBorder(src, top, bottom, left, right, borderType, value)

Src : *It is the source image.*
Top : *It is the border width in number of pixels in top direction.*
Bottom : *It is the border width in number of pixels in bottom direction.*
Left : *It is the border width in number of pixels in left direction.*
Right : *It is the border width in number of pixels in right direction.*
borderType : *It depicts what kind of border to be added. It is defined by flags like* ***cv2.BORDER_CONSTANT****,* ***cv2.BORDER_REFLECT****, etc.*

value : *It is an optional parameter which depicts color of border if border type is* ***cv2.BORDER_CONSTANT****.*

Borders are of following types :

cv2.BORDER_CONSTANT: *It adds a constant colored border. The value should be given as a keyword argument*

cv2.BORDER_REFLECT: *The border will be mirror reflection of the border elements*

cv2.BORDER_REFLECT_101 *or* ***cv2.BORDER_DEFAULT:*** *It does the same works as* ***cv2.BORDER_REFLECT*** *but with slight change.*

cv2.BORDER_REPLICATE: *It replicates the last element. Suppose, if image contains letters "**abcdefgh**" then output will be "**aaaaa|abcdefgh|hhhhh**".*

Splitting and Merging channels

To do this, we use **cv2.split() and cv2.merge()** functions respectively

Image ROI (Region of Interest)

ROI is used where we will have to play with certain region of images

For e.g. :Eye detection in images

This approach improves accuracy (because eyes are always on faces) and performance (because we search for a small area).

ROI is obtained using Numpy indexing

Arithmetic and bitwise operation on image

Arithmetic Operations like Addition, Subtraction, and Bitwise Operations(AND, OR, NOT, XOR) can be applied to the input images.

These operations can be helpful in enhancing the properties of the input images.

We can add two images by using function cv2.add()