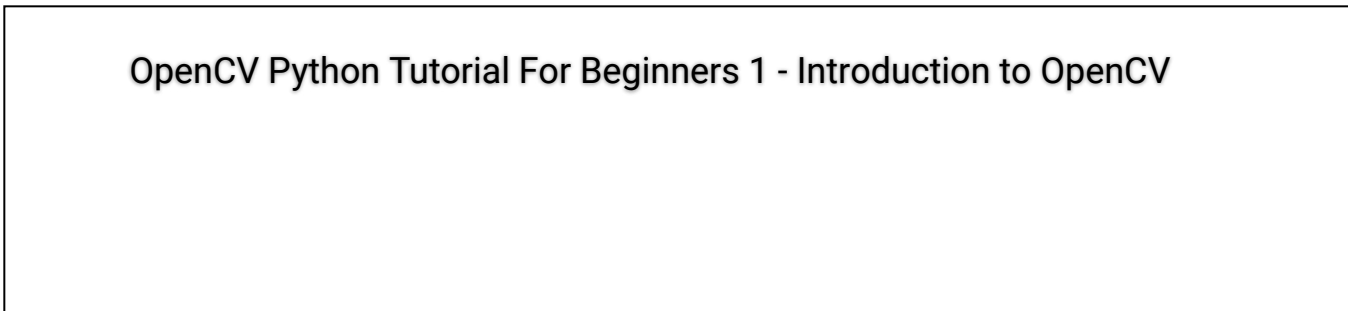
OpenCV Tutorial

In this tutorial, you are going to learn about OpenCV library and the functionalities it provides. OpenCV is a large open-source library for computer vision, machine learning, and image processing, and it currently plays a critical part in real-time operations, which are critical in today's systems. It may be used to detect items, faces, and even human handwriting in photos and movies. Python can process the OpenCV array structure for analysis when it is combined with other modules such as NumPy. We employ vector space and execute mathematical operations on these features to identify visual patterns and their various features.

Watch the video given below.

```
###Refer Video
from IPython.display import YouTubeVideo
YouTubeVideo('kdLM6AOd2vc', width=700, height=400)
```

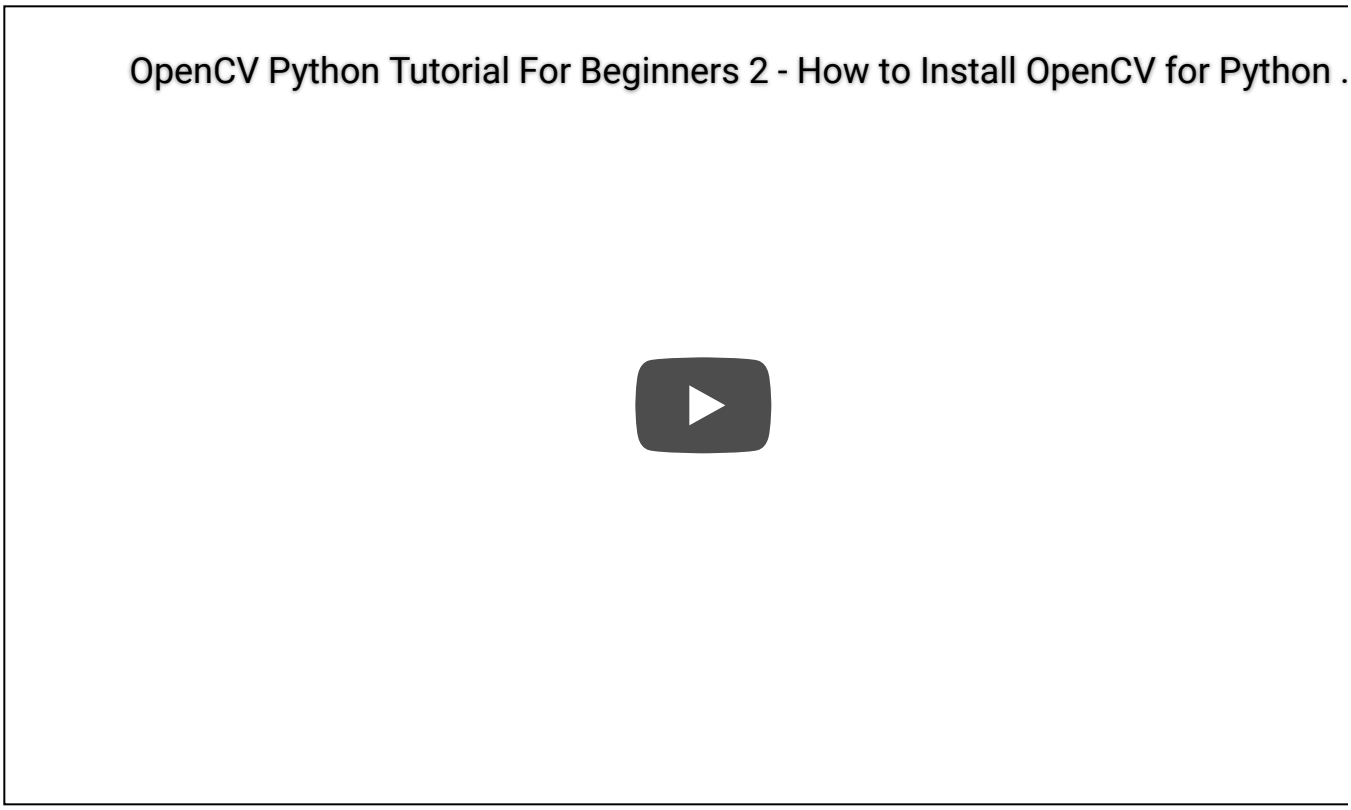


Note: Implement the tutorial in Jupyter notebook.

Install the open-cv library

Let's start by installing the open CV library

```
YouTubeVideo('d3AT9EGp4iw', width=700, height=400)
```



Installing OpenCV

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

```
pip install opencv-python
```

Pypi Link of opencv-contrib

[Pypi link of opencv-python]

```
# install the library
```

```
Requirement already satisfied: opencv-python in c:\users\windos\anaconda3\lib\site-packa
Requirement already satisfied: numpy>=1.17.3 in c:\users\windos\anaconda3\lib\site-packa
```

python requirements version>=3.6

OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:-

- Core functionality (core) - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
- Image Processing (imgproc) - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
- Video Analysis (video) - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
- Camera Calibration and 3D Reconstruction (calib3d) - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
- 2D Features Framework (features2d) - salient feature detectors, descriptors, and descriptor matchers.
- Object Detection (objdetect) - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
- High-level GUI (highgui) - an easy-to-use interface to simple UI capabilities.
- Video I/O (videoio) - an easy-to-use interface to video capturing and video codecs.
- ... some other helper modules, such as FLANN and Google test wrappers, Python bindings, and others.

Importing the library

```
#import cv2
```

Loading the image

Use any of your favorite image in this tutorial or you can simply use tha image given to you.

```
#get the image path
```

Refer the video given below before starting the implementation

```
### Refer video
YouTubeVideo('TGQcDaZ56ao', width=700, height=400)
```

Reading the image

The imread() function loads image from the specified file and returns it. It opens a file and loads an image from it. This method produces an empty matrix if the picture cannot be read (due to a missing file, poor permissions, or an unsupported or invalid format).

Arguments of the function : -

- cv2.IMREAD_COLOR : Loads a color image. Any transparency of image will be neglected. It is the default flag.

- cv2.IMREAD_GRAYSCALE : Loads image in grayscale mode
- cv2.IMREAD_UNCHANGED : Loads image as such including alpha channel Instead of these three flags, you can simply pass integers 1, 0 or -1 respectively.

Refer: https://docs.opencv.org/4.x/d4/da8/group__imgcodecs.html

```
# image will contain 3 channels by default
# 0 denotes read as  grayscale image
# Loading an color image in rgb
# Loading an color image in rgb
```

Displaying the image

cv2.imshow() method is used to display an image in a window. The window automatically fits to the image size.

Refer: https://www.geeksforgeeks.org/python-opencv-cv2-imshow-method/

cv2.waitKey() is a keyboard binding function. A call to waitKey() starts a message passing cycle that waits for a key stroke in the "image" window. Its argument is the time in milliseconds. The function waits for specified milliseconds for any keyboard event. If you press any key in that time, the program continues. If 0 is passed, it waits indefinitely for a key stroke.
https://www.geeksforgeeks.org/python-opencv-waitkey-function/

cv2.destroyAllWindows() simply destroys all the windows we created. It destroyAllWindows() is required to close all the popping windows. If you want to destroy any specific window, use the function cv2.destroyWindow() where you pass the exact window name as the argument.
https://www.geeksforgeeks.org/python-opencv-destroyallwindows-function/

```
# imshow() displays the image.
#display the image in grayscale

#set waitkey

#destroy all windows

# A call to waitKey() starts a message passing cycle that waits for a key stroke in the "imag
# destroyAllWindows() is required to close all the popping windows.


#show the RGB image
 # imshow() displays the image.
```

```
# A call to waitKey() starts a message passing cycle that waits for a key stroke in the "imag
# destroyAllWindows() is required to close all the popping windows.
```

Writing or saving the image

cv2.imwrite() is used to save the image.

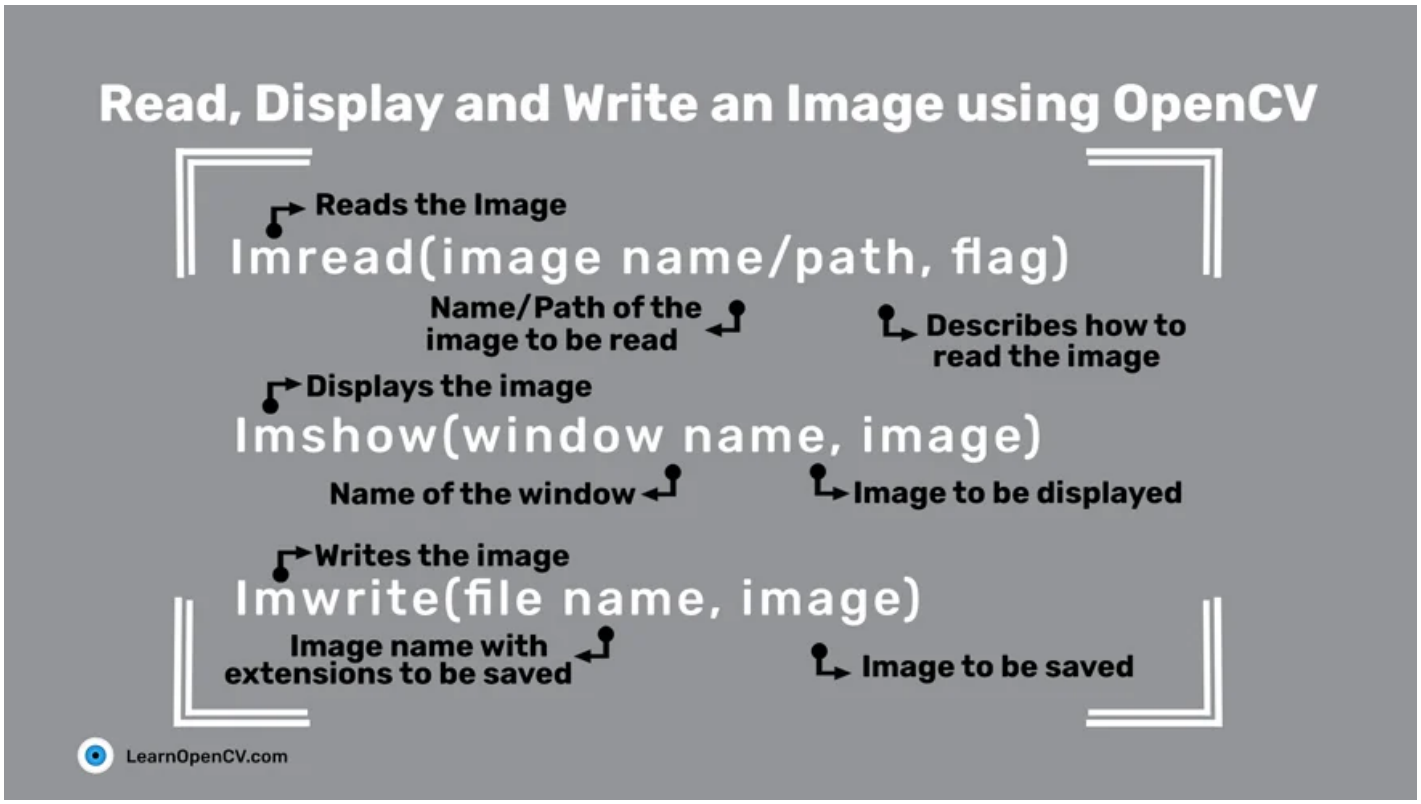
- First argument is the file name
- second argument is the image you want to save.

Refer:https://www.geeksforgeeks.org/python-opencv-cv2-imwrite-method/

```
#save the original image
```

```
True
```

For reference:



Matplotlib for image processing

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats .
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

Install Matplotlib

Matplotlib can be installed using pip. The following command is run in the command prompt to install Matplotlib.

pip install matplotlib

```
#install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\windos\anaconda3\lib\site-packages
Requirement already satisfied: cycler>=0.10 in c:\users\windos\anaconda3\lib\site-packag
Requirement already satisfied: pillow>=6.2.0 in c:\users\windos\anaconda3\lib\site-packa
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\winc
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\windos\anaconda3\lib\site-p
Requirement already satisfied: python-dateutil>=2.1 in c:\users\windos\anaconda3\lib\sit
Requirement already satisfied: certifi>=2020.06.20 in c:\users\windos\anaconda3\lib\site
Requirement already satisfied: numpy>=1.15 in c:\users\windos\anaconda3\lib\site-package
Requirement already satisfied: six in c:\users\windos\anaconda3\lib\site-packages (from
```

importing matplotlib

```
#import matplotlib
```

Reading and Displaying the image

Below we read and display an image using matplotlib. The imshow method of matplotlib is used to do so. Refer:https://www.geeksforgeeks.org/working-with-images-in-python-using-matplotlib/#:~:text=The%20image%20module%20in%20matplotlib,used%20to%20display%20the%20image.

```
#import numpy,cv2 and matplotlib


#read the saved image

#show image using plt.imshow

# to hide tick values on X and Y axis
```

Very important note:-

1. Color image loaded by OpenCV is in BGR mode.
2. But Matplotlib displays in RGB mode.
3. So color images will not be displayed correctly in Matplotlib if image is read with OpenCV.

Properties of image

In this section we will look at some of the properties of images like shape,pixels,datatype,height and width. Start by reading an image.

```
#import cv2 and numpy


# Load the color image
```

Shape

Images are stored in numpy ndarray when using OpenCV Python. Use ndarray.shape to get the dimensions of an image's shape or size. Then, for each pixel, you can use index on the dimensions variable to get the width, height, and number of channels. Shape of image is accessed by img.shape. It returns a tuple of number of rows, columns and channels (if image is color).

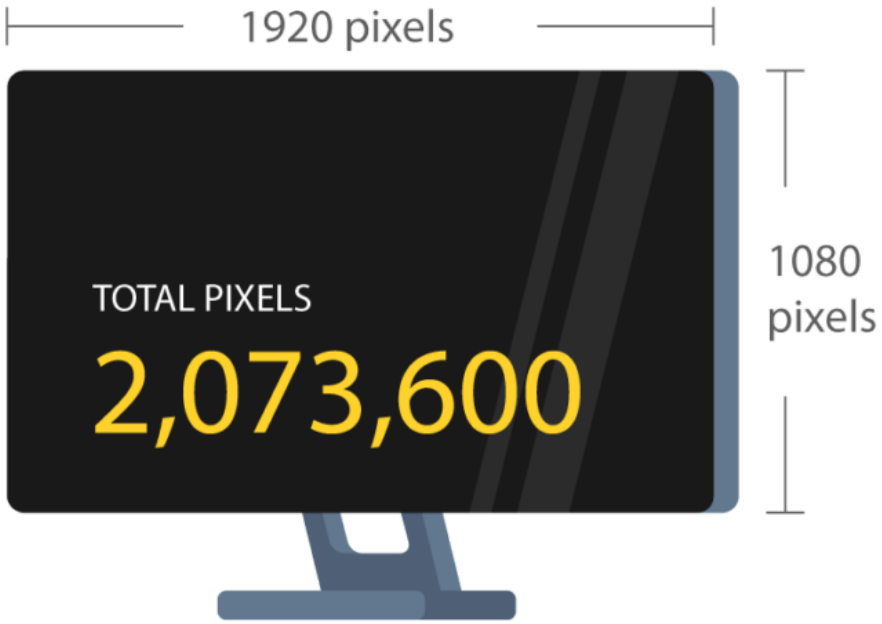
Refer: https://www.tutorialkart.com/opencv/python/opencv-python-get-image-size/

```
#display shape of image
# it has 3 channels i.e RGB image
```

```
(360, 360, 3)
```

Total Pixels

Total number of pixels is accessed by img.size:




```
#display number of image pixels
```

```
388800
```

Datatype

Image datatype is obtained by img.dtype

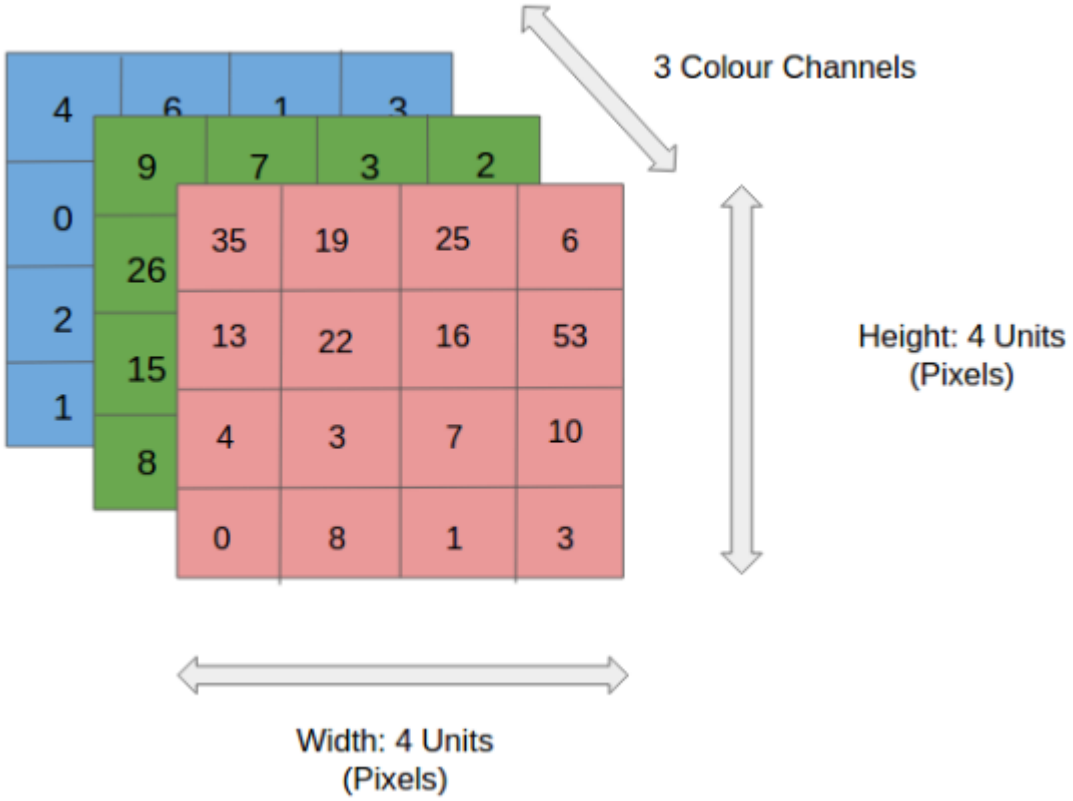
Refer: https://scikit-image.org/docs/dev/user_guide/data_types.html

```
#display image datatype
```

```
uint8
```

Finding height , width and number of channels

height is the 0th value of image shape,width is the 1st value and 2nd value denotes number of channels.




```
# height, width, number of channels in image
```

```
#print image height
```

```
#print width
```

```
#print number of channels
```

```
Image Height       :  360
Image Width        :  360
Number of Channels :  3
```

cvtColor opencv

The cvtColor is used to convert an image from one color space to another. The syntax is following:

cv2.cvtColor(src, dst, code)

Refer: https://www.geeksforgeeks.org/python-opencv-cv2-cvtcolor-method/

Parameters:
src - It is used to input an image: 8-bit unsigned.

dst - It is used to display an image as output. The output image will be same size and depth as input image.

code - color space conversion code.

```
# Read an image in default mode

# set Window name in which image is displayed

# Using cv2.cvtColor() method
# Using cv2.COLOR_BGR2GRAY color space for convert BGR image to grayscale

# convert image to grayscale

# Displaying the image

#set waitkey

#destroy all windows
```

This is the type of output that you will obtain in your system.

Applying some filters

In the section, you will adding some interesting filters to the images. Start by loading the original color image. Create a kernel of size 5,5 uisng numpy.ones function. This kernel will be useful in applying various filters.

```
#import cv2 and numpy


# Creating a 5x5 matrix kerne lwith each cell having value 1
#use np.ones to do so
#kernel = np.ones((5,5),np.uint8)

#set path of original color image

#read the image path


#show the image read above
        #"REAL" is the window name

#set wait key

#destroy all windows
```

This is the type of output that you will obtain in your system.

Blur filter

The image is blurred using the GaussianBlur method of cv2. We should specify the width and height of the kernel which should be positive and odd. We also should specify the standard deviation in the X and Y directions, sigmaX and sigmaY respectively.

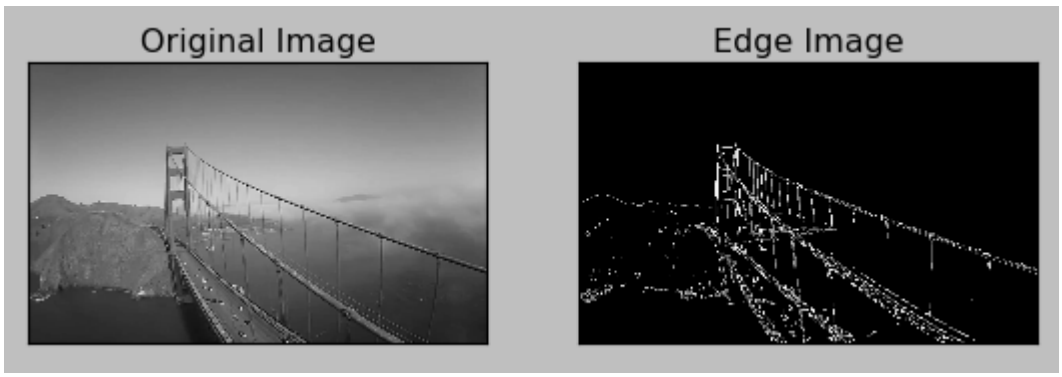
Refer:https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html

```
#apply GaussianBlur to the original image

#show the blurred image
              #"BLUR" is the window name
#set wait key

#destroy all windows
```

This is the type of output that you will obtain in your system.

Canny filter

Canny filter highlights only the edges of the image. This is acheived by Canny method of cv2. First argument is our input image. Second and third arguments are our minVal and maxVal respectively. Fourth argument is aperture_size. It is the size of Sobel kernel used for find image gradients. By default it is 3. Last argument is L2gradient which specifies the equation for finding gradient magnitude. If it is True, it uses the equation mentioned above which is more accurate, otherwise it uses this function: Edge_Gradient(G)=|Gx|+|Gy|. By default, it is False.



Refer:https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html

```
#apply Canny to blurred image


#show the image with canny filter
                      #"EDGES" is the window name
```

```
#set wait key

#destroy all windows
```

This is the type of output that you will obtain in your system.

Grayscaling filter

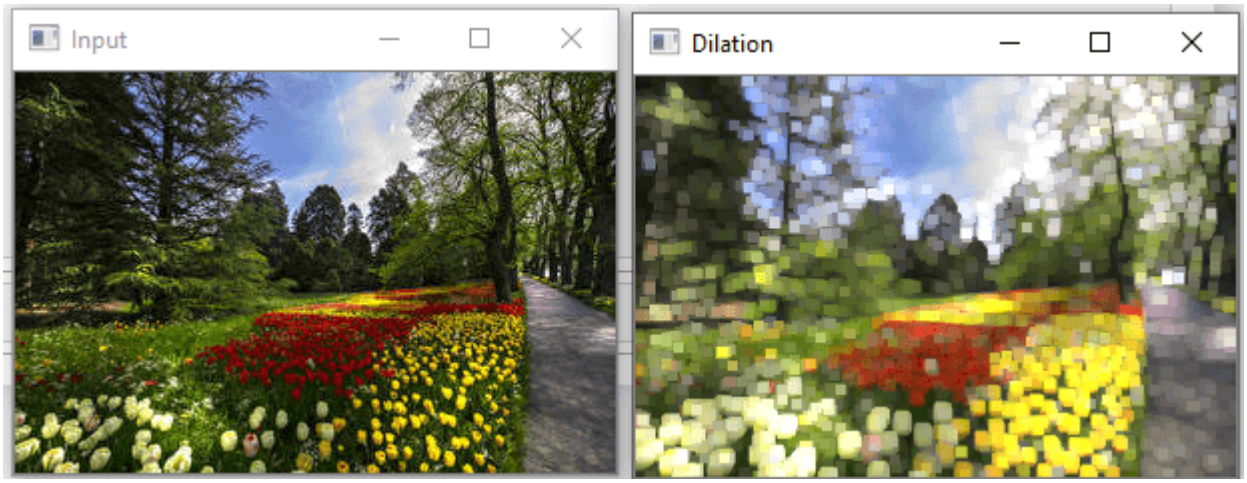
It displays the image in grayscale. This can be done by using cvtColor method and pass the image and cv2.COLOR_BGR2GRAY into it.

Refer: https://www.geeksforgeeks.org/python-opencv-cv2-cvtcolor-method/

```
#apply cvtColor to the original image

#show the gray image
                    #"GRAY" is the window name

#set wait key

#destroy all windows
```

Pixel dilation filter

It is done by using dilate method of cv2. Use this method on the canny filtered image.



Working of dilation:

- A kernel(a matrix of odd size(3,5,7) is convolved with the image
- A pixel element in the original image is '1' if at least one pixel under the kernel is '1'.
- It increases the white region in the image or the size of the foreground object increases

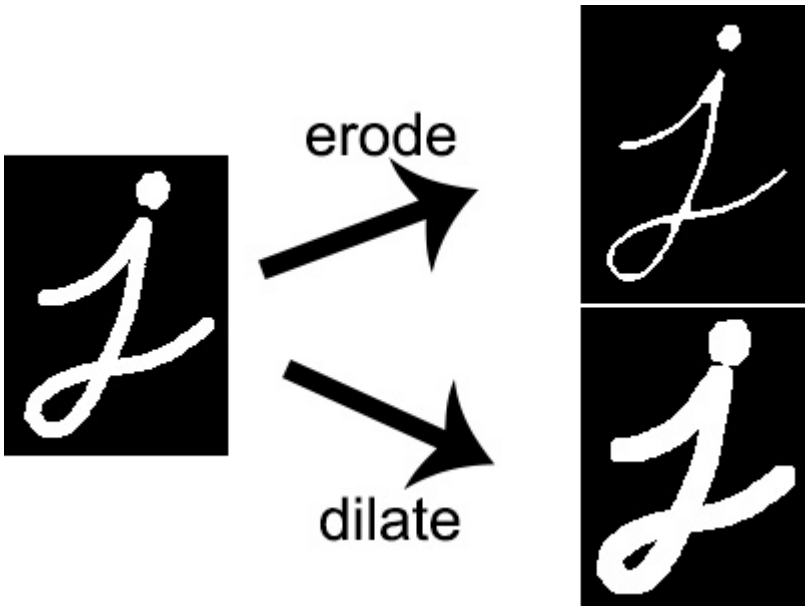
https://www.geeksforgeeks.org/erosion-dilation-images-using-opencv-python/

```
#apply dilate to canny filtered image

#show the pixel dilated image
                      #"DILATE" is the window name

#set wait key

#destroy all windows
```

This is the type of output that you will obtain in your system.

Pixel erosion filter

Use erode method of cv2 passing the original image.



Working of erosion:

- A kernel(a matrix of odd size(3,5,7) is convolved with the image.
- A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel are 1, otherwise, it is eroded (made to zero).
- Thus all the pixels near the boundary will be discarded depending upon the size of the kernel.

- So the thickness or size of the foreground object decreases or simply the white region decreases in the image.

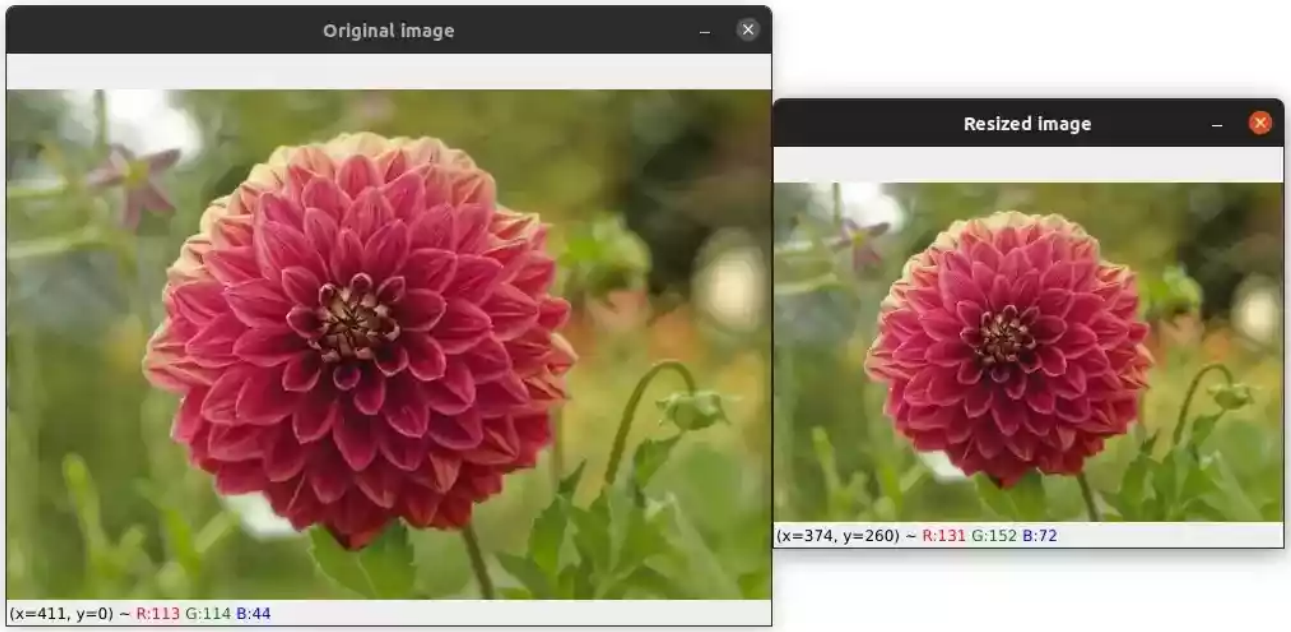
https://www.geeksforgeeks.org/erosion-dilation-images-using-opencv-python/

```
#apply erode to the original image

#show the eroded image
                #"ERODE" is the window name

#set wait key

#destroy all windows
```

This is the type of output that you will obtain in your system.

Image Resizing

This part applies image resizing that is changes the height and width of the images using the resize method.



Refer:https://www.geeksforgeeks.org/image-resizing-using-opencv-python/

```
#import cv2

#read the original color image

#read the same original color image in another variable

#set scale variable to 40 (reduces size by 40%)


#get the new width
#int(resized_img.shape[1] * scale / 100)
                                  # reducing the image width by scale percentage(here 40%)
#get the new height
                                  # reducing the image height by scale percentage(here 40%)
#store the new width and height in a tuple
# resize image
#cv2.resize(resized_img, dim, interpolation=cv2.INTER_AREA)

#print original dimensions
                                # it has 3 Channels i.e RGB image
#print resized dimensions


#show the resized image
                               #"Resized image" is the window name
#set wait key
```

```
#destroy all windows
```

```
Original Dimensions:  (360, 360, 3)
Resized Dimensions :  (144, 144, 3)
```

Image Rotation

Image rotation is achieved by using rotate function of cv2. cv2.rotate() method is used to rotate a 2D array in multiples of 90 degrees. The function cv::rotate rotates the array in three different ways.

Refer:https://www.geeksforgeeks.org/python-opencv-cv2-rotate-method/

```
#import cv2 and numpy



#read the original color image 3 times in different variables
#read the original color image

#read the same original color image in another variable

#read the same original color image in another variable


#rotate the image counter clockwise
#cv2.rotate(img_ori1, cv2.ROTATE_90_COUNTERCLOCKWISE)

#rotate the image clockwise
#cv2.rotate(img_ori1, cv2.ROTATE_90_CLOCKWISE)


#show the original image

#show the counter clockwise rotated image

#show the clockwise rotated image

#set wait key

#destroy all windows
```

This is the type of output that you will obtain in your system.

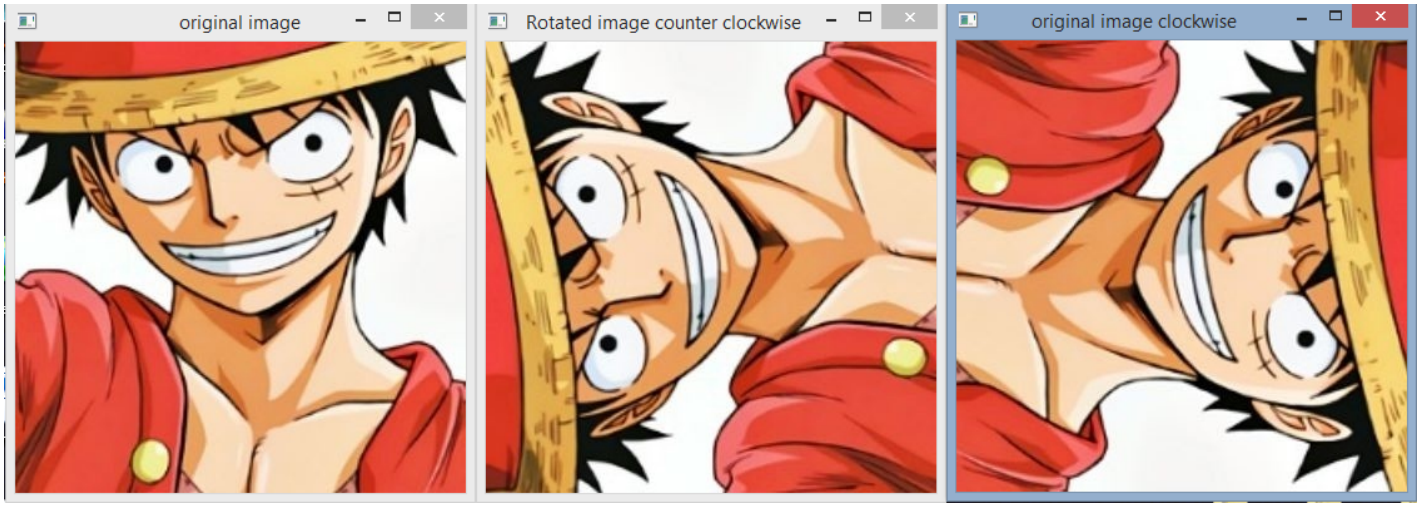


Image Scaling

The scaling of images is referred to as image resizing. Scaling is useful in a variety of image processing and machine learning applications. It aids in the reduction of the number of pixels in an image, which has various advantages, for example. It can shorten the time it takes to train a neural network since the more pixels in an image there are, the more input nodes there are, which raises the model's complexity.

Refer:https://www.geeksforgeeks.org/image-resizing-using-opencv-python/

```
#import cv2 and numpy


#read the original color image


#scale up in X direction
#scaled_up_x=cv2.resize(img_ori,None, fx=2,fy=1, interpolation=cv2.INTER_CUBIC)

#scale down in X direction
#scaled_down_x=cv2.resize(img_ori,None, fx=0.5,fy=1, interpolation=cv2.INTER_LINEAR)

#scale up in Y direction
#scaled_up_y=cv2.resize(img_ori,None, fx=1,fy=2, interpolation=cv2.INTER_CUBIC)

#scale down in Y direction
#scaled_down_y=cv2.resize(img_ori,None, fx=1,fy=0.5, interpolation=cv2.INTER_LINEAR)

#show all the scaled images


#set wait key
```

```
#destroy all windows
```

This is the type of output that you will obtain in your system.

Image Cropping

In this section we will implement image cropping. Every image that is read in, gets stored in a 2D array (for each color channel). Simply specify the height and width (in pixels) of the area to be cropped. And it's done!

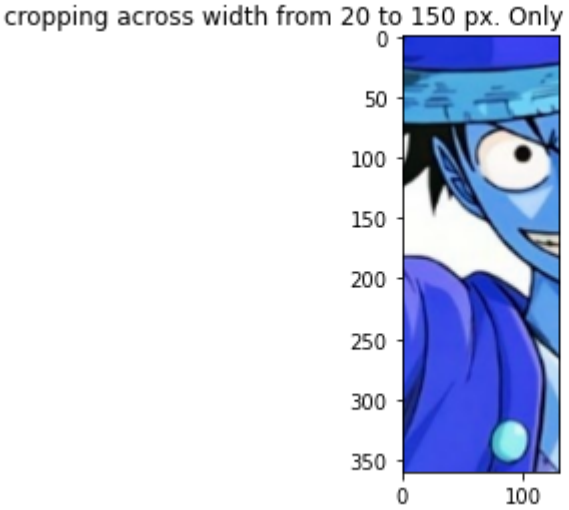
Refer: https://learnopencv.com/cropping-an-image-using-opencv/

```
#read the original image

#show the image using matplotlib imshow

#set the title
```

```
Text(0.5, 1.0, 'Original Image')
```


```
#import cv2,matplotlib.pyplot,numpy


#read the original image

#crop the image by reducing the width
#width_crop_1 = img[:,20:150] (cropping across width from 20 to 150 px. Only width from 20 to

#show the cropped image

#set the title
```

```
Text(0.5, 1.0, 'cropping across width from 20 to 150 px. Only width from 20 to 150 px is
```



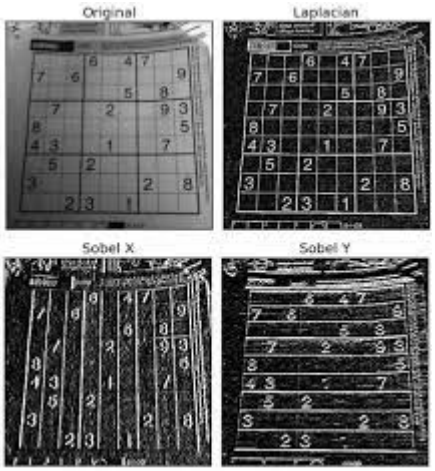

```
#import cv2,matplotlib.pyplot,numpy


#read the original image

#crop the image by reducing height
#height_crop_1 = img[10:180,:] (cropping across height from 10 to 180 px. Only height from 10

#show the cropped image

#set title
```

```
Text(0.5, 1.0, 'cropping across height from 10 to 180 px. Only height from 10 to 180 px
```

Edge Detection technique (Sobel Method)

Mathematical methods are used to discover points in an image where the brightness of pixel intensities fluctuates noticeably.

Refer: https://www.geeksforgeeks.org/python-program-to-detect-the-edges-of-an-image-using-opencv-sobel-edge-detection/



Sobel in X direction

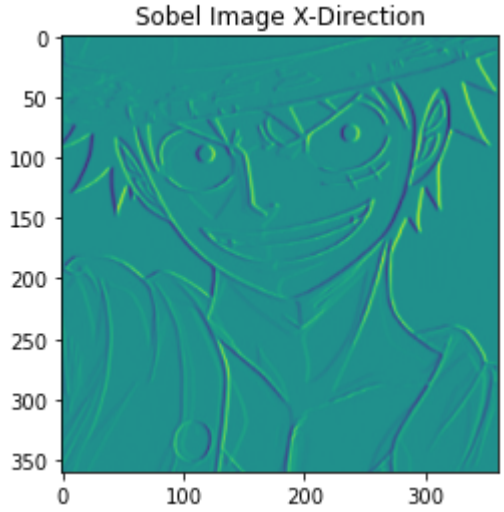
It is obtained through the convolution of the image with a matrix called kernel which has always odd size. The kernel with size 3 is the simplest case.

we can't use original RGB image as we are using Matplotlib. We have to convert it to GRAY image

```
#import cv2,matplotlib.pyplot and numpy
```

```
#read the original image


# we can't use original RGB image as we are using Matplotlib
#convert to grayscale
                                  # original image converted to gray image using cvtColor
#apply sobel in X direction
                                  # 1,0 for sobel in x axis

# Plotting the sobel image
```

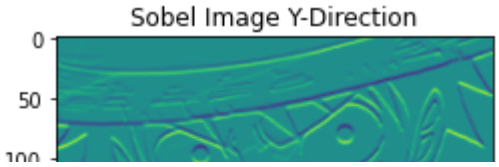
```
Text(0.5, 1.0, 'Sobel Image X-Direction')
```



Sobel in Y direction

```
#sobel in Y direction
#cv2.Sobel(gray_image, cv2.CV_32F , 0, 1, ksize=1)
                                                    #and use 0,1 for sobel in y axis
#show image

#set title
```

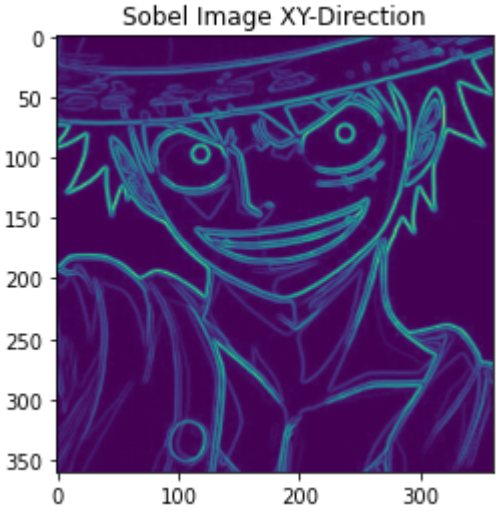
```
Text(0.5, 1.0, 'Sobel Image Y-Direction')
```



Sobel in both X and Y direction


```
#sobel in x and y direction
#nm.sqrt(sobel_img_y**2+sobel_img_x**2)

#show the image

#set title
```

```
Text(0.5, 1.0, 'Sobel Image XY-Direction')
```



OpenCv for Video Processing

In this section,we aim to use open CV for video processing.

use cv2.VideoCapture(0) to capture video from live camera

use cv2.VideoCapture('filename') to capture saved videos from local disk

https://docs.opencv.org/4.x/dd/d43/tutorial_py_video_display.html

```
### Refer video
YouTubeVideo('-RtVZsCvXAQ', width=700, height=400)
```

Capturing videos from live camera

A VideoCapture object is required to capture a video. The device index or the name of a video file can be used as a parameter. A device index is just a number that identifies which camera is being used. Normally, only one camera is connected (as in my case). As a result, I simply pass 0 (or -1). By passing 1 and so on, you can select the second camera. After that, you can capture each frame individually. But don't forget to release the capture at the conclusion.

PRESS q to close the pop up window

```
#import cv2 and numpy


#capture video from camera


#while(True)

    # Capture image frame-by-frame
    #ret, frame = cap.read()

    # Display the resulting frame

    #if cv2.waitKey(1) & 0xFF == ord('q'):
```

```
                    # PRESS q to close the pop up window

# When everything done, release the capture using release method

#destroy all windows
```

Capturing videos from local disks

Make sure to download a video in your device. Or you can simply use the given video.

Playing video from a file is the same as capturing it from a camera; the only difference is that the camera index is replaced with the name of the video file. Use proper time for cv.waitKey while displaying the frame (). If it's too low, the video will be extremely fast, and if it's too high, the video will be extremely slow (Well, that is how you can display videos in slow motion). In most circumstances, 25 milliseconds will suffice.

PRESS q to close the pop up window

```
#import cv2 and numpy


#use VideoCapture,pass the filename

#while(cap.isOpened()):

    #read the captures
    #ret, frame = cap.read()

    #you can also open the camera in the grayscale mode
    #gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    #cv2.imshow('frame',gray)

    #show the frame

    #set waitkey as 1 and 'q' for closing window

# When everything done, release the capture

#destroy all windows
```

Congratulations!!! You've done it.

In this assignment you learned:

- OpenCV
- Processing images and videos with OpenCV

Keep practising!!

Do fill the feedback form given below:

[Feedback form]