**DAY 2:OPENCV COMMANDS AND FUNCTIONS**

**#READING AN IMAGE**

import cv2 as cv

img=cv.imread('dog1.jpg')

cv.imshow('Dog',img)

**#grayscale**

gray=cv.cvtColor(img,cv.COLOR\_BGR2GRAY)

cv.imshow('Gray',gray)

**#blur**

blur=cv.GaussianBlur(img,(3,3),cv.BORDER\_DEFAULT)

cv.imshow('Blur',blur)

**#RESIZE**

def rescaleframe(frame,scale=5):

    width=frame.shape[1]\*scale

    height=frame.shape[0]\*scale

    dimensions=(width,height)

    return rescaleframe(frame,dimensions,interpolation=cv.INTER\_AREA)

img=cv.imread('dog2.jpg')

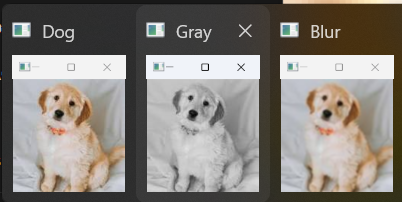
while True:

    isTrue,frame=img.read()

    frameresized=rescaleframe(frame)

    cv.imshow('Resized image',frameresized)

cv.waitKey(0)



Resized image:



**WRITING AN IMAGE INTO THE DIRECTORY:**

import cv2

import os

image\_path=r'C:\Users\Sanjana Raghunath\OneDrive\Documents\openCV\cat3.jpg'

directory=r'C:\Users\Sanjana Raghunath\OneDrive\Documents\openCV'

image=cv2.imread(image\_path)

os.chdir(directory)

print("Before saving image:")

print(os.listdir(directory))

filename='newimage.jpg'

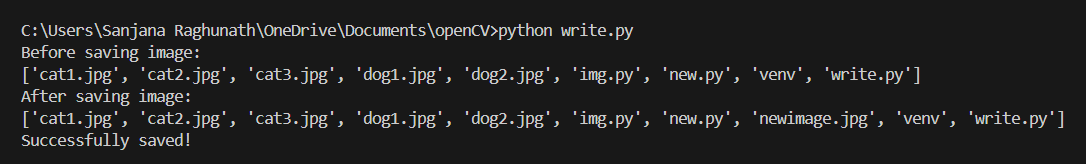
cv2.imwrite(filename,image)

print("After saving image:")

print(os.listdir(directory))

print("Successfully saved!")

OUTPUT:



**BGR SPLIT IN AN IMAGE:**

import cv2

image=cv2.imread('dog2.jpg')

B,G,R=cv2.split(image)

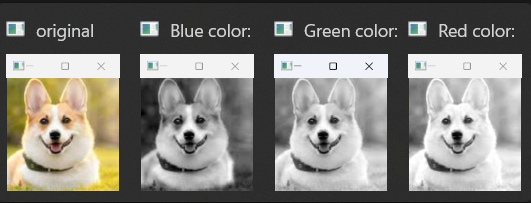
cv2.imshow("original",image)

cv2.imshow('Blue color:',B)

cv2.imshow('Green color:',G)

cv2.imshow('Red color:',R)

cv2.waitKey(0)



**ARITHMETIC OPERATIONS USING OPENCV:**

ADDITION:

import cv2

import numpy as np

img1=cv2.imread('dog1.jpg')

img2=cv2.imread('cat3.jpg')

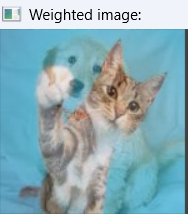
img2=cv2.resize(img2,(img1.shape[1],img1.shape[0]))

weightedsum=cv2.addWeighted(img1,0.5,img2,0.5,0)

cv2.imshow('Weighted image:',weightedsum)

if cv2.waitKey(0) or 0xff==27:

    cv2.destroyAllWindows()



Subtraction:

import cv2

import numpy as np

img1=cv2.imread('img1.png')

img2=cv2.imread('img2.png')

img2=cv2.resize(img2,(img1.shape[1],img1.shape[0]))

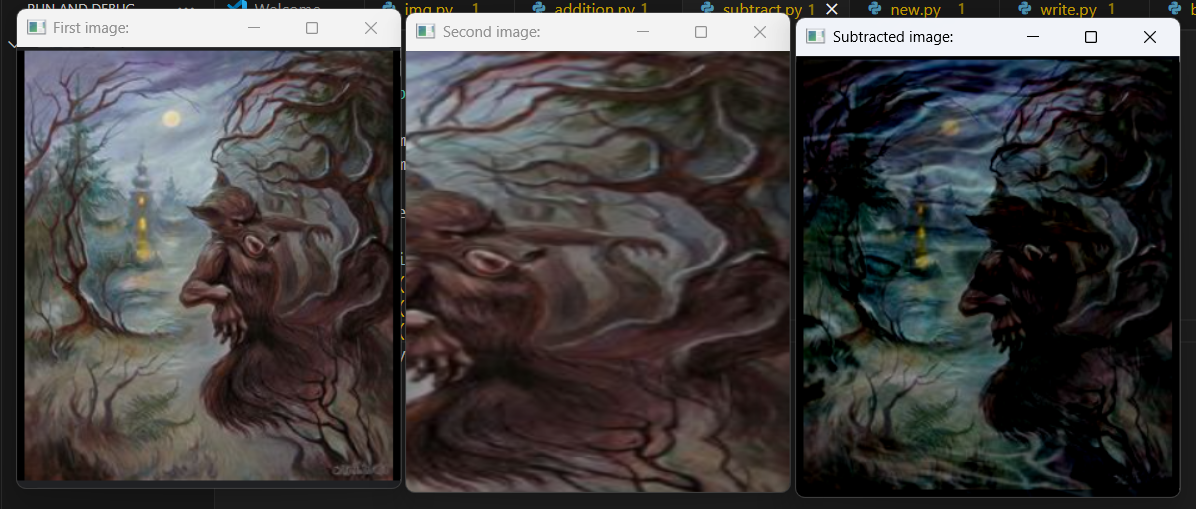
subtractedimage=cv2.subtract(img1,img2)

cv2.imshow('First image:',img1)

cv2.imshow('Second image:',img2)

cv2.imshow('Subtracted image:',subtractedimage)

cv2.waitKey(0)



BITWISE OPERATIONS:

import cv2

#AND

import numpy as np

and1=cv2.imread('and1.png')

and2=cv2.imread('and2.png')

and2=cv2.resize(and2,(and1.shape[1],and1.shape[0]))

andimage=cv2.bitwise\_and(and1,and2,mask=None)

cv2.imshow('First image:',and1)

cv2.imshow('Second image:',and2)

cv2.imshow('And image:',andimage)

#OR operator image

and1=cv2.imread('and1.png')

and2=cv2.imread('and2.png')

and2=cv2.resize(and2,(and1.shape[1],and1.shape[0]))

orimage=cv2.bitwise\_or(and1,and2,mask=None)

cv2.imshow('First image:',and1)

cv2.imshow('Second image:',and2)

cv2.imshow('or image:',orimage)

#xor

and1=cv2.imread('and1.png')

and2=cv2.imread('and2.png')

and2=cv2.resize(and2,(and1.shape[1],and1.shape[0]))

xorimage=cv2.bitwise\_xor(and1,and2,mask=None)

cv2.imshow('First image:',and1)

cv2.imshow('Second image:',and2)

cv2.imshow('xor image:',xorimage)

# not

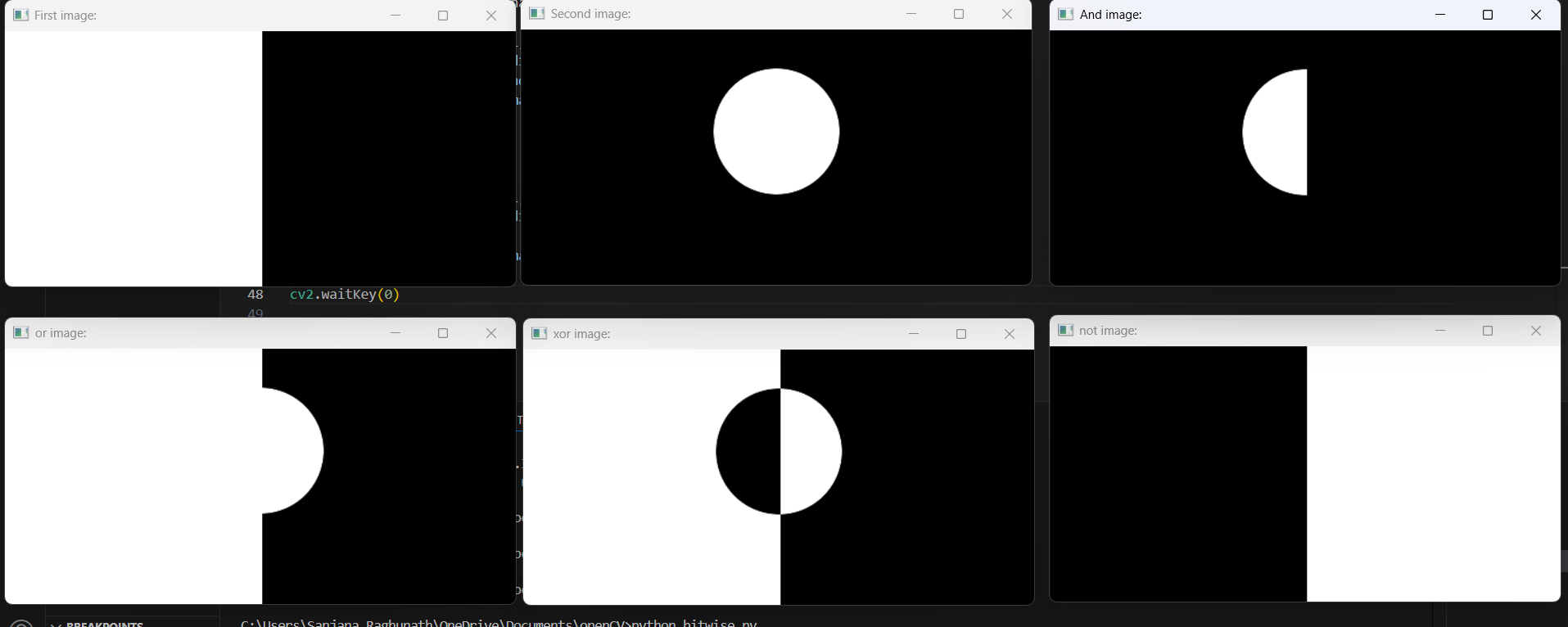
and1=cv2.imread('and1.png')

notimage=cv2.bitwise\_not(and1,mask=None)

cv2.imshow('First image:',and1)

cv2.imshow('not image:',notimage)

OUTPUT:



Erode()

import cv2

import numpy as np

path = r'C:\Users\Sanjana Raghunath\OneDrive\Documents\openCV\cat2.jpg'

image = cv2.imread(path)

window\_name = 'Image'

kernel = np.ones((5, 5), np.uint8)

image = cv2.erode(image, kernel)

cv2.imshow(window\_name, image)

cv2.waitKey(0)

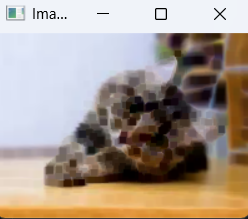


Image border:

import cv2

path = r'C:\Users\Sanjana Raghunath\OneDrive\Documents\openCV\dog1.jpg'

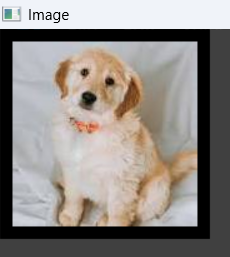
image = cv2.imread(path)

window\_name = 'Image'

image = cv2.copyMakeBorder(image, 10, 10, 10, 10, cv2.BORDER\_CONSTANT, None, value = 0)

cv2.imshow(window\_name, image)

cv2.waitKey(0)



Grayscale :

import cv2

img = cv2.imread(r'C:\Users\Sanjana Raghunath\OneDrive\Documents\openCV\dog1.jpg')

(row, col) = img.shape[0:2]

for i in range(row):

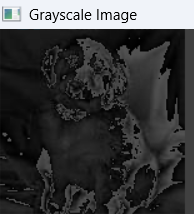
    for j in range(col):

        img[i, j] = sum(img[i, j]) \* 0.33

cv2.imshow('Grayscale Image', img)

cv2.waitKey(0)

cv2.destroyAllWindows()



EROSION AND DILATION:

import cv2

import numpy as np

img = cv2.imread('img1.jpg')

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

\_, thresh = cv2.threshold(gray, 120, 255, cv2.THRESH\_BINARY)

kernel = np.ones((5, 5), np.uint8)

eroded = cv2.erode(thresh, kernel, iterations=1)

dilated = cv2.dilate(thresh, kernel, iterations=1)

cv2.imshow('Original', img)

cv2.imshow('Thresholded', thresh)

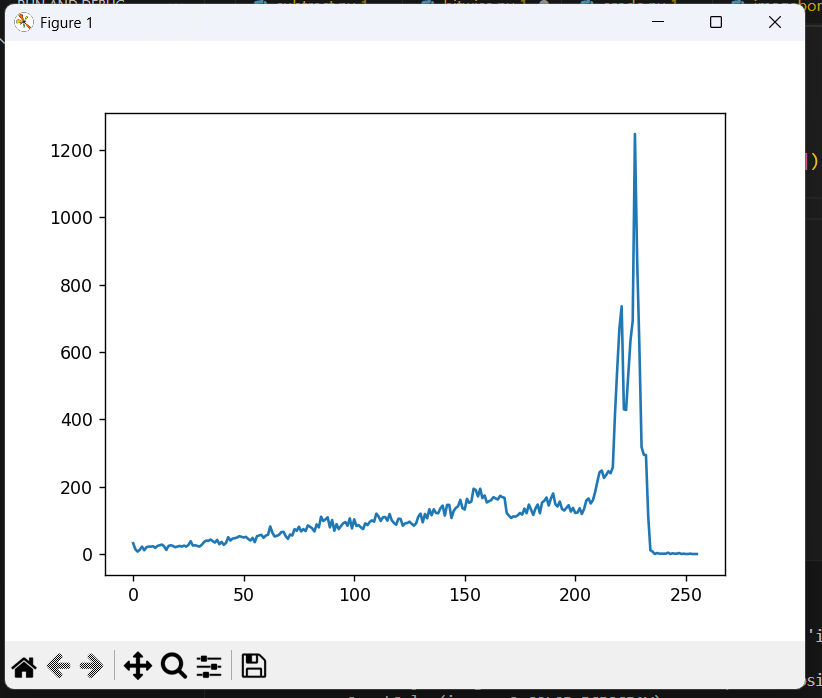
cv2.imshow('Eroded', eroded)

cv2.imshow('Dilated', dilated)

cv2.waitKey(0)

cv2.destroyAllWindows()

**histogram:**



import cv2

from matplotlib import pyplot as plt

img = cv2.imread('cat2.jpg', 0)

histr = cv2.calcHist([img], [0], None, [256], [0, 256])

plt.plot(histr)

plt.show()

**histogram equalization:**

import cv2

from matplotlib import pyplot as plt

img = cv2.imread('cat2.jpg', 0)

equalized = cv2.equalizeHist(img)

cv2.imshow('Original', img)

cv2.imshow('Equalized', equalized)

cv2.waitKey(0)

cv2.destroyAllWindows()

plt.figure()

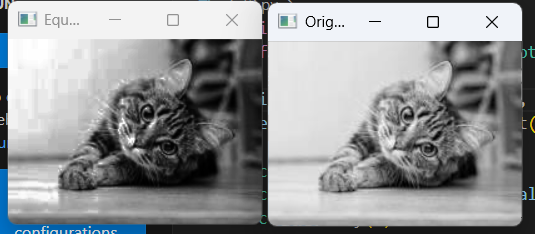
plt.title("Histogram Comparison")

plt.plot(cv2.calcHist([img], [0], None, [256], [0, 256]), label='Original')

plt.plot(cv2.calcHist([equalized], [0], None, [256], [0, 256]), label='Equalized')

plt.legend()

plt.show()



**Spectral image map:**

import matplotlib.pyplot as plt

import cv2

img = cv2.imread('cat3.jpg')

plt.imshow(img, cmap ='nipy\_spectral')

cv2.waitKey(0)

**BILATERAL FILTERING:**

OpenCV has a function called **bilateralFilter()** with the following arguments: 

1. **d:** Diameter of each pixel neighborhood.
2. **sigmaColor:** Value of   *σ*  in the color space. The greater the value, the colors farther to each other will start to get mixed.
3. **sigmaSpace:** Value of *σ*  in the coordinate space. The greater its value, the more further pixels will mix together, given that their colors lie within the sigmaColor range.

import cv2

img = cv2.imread('scenery.jpg')

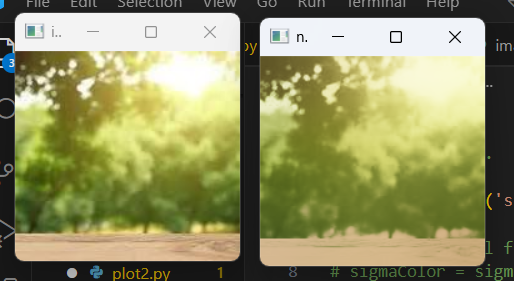
bilateral = cv2.bilateralFilter(img, 15, 75, 75)

cv2.imwrite('newscenery.jpg', bilateral)

cv2.imshow('image',img)

cv2.imshow('new clear image:',bilateral)

OUTPUT:



**IMAGE INPAINTING:**

I**mage inpainting** is the process of removing damage, such as noises, strokes or text, on images. It is particularly useful in the restoration of old photographs which might have scratched edges or ink spots on them.

Code:

import cv2

import numpy as np

img = cv2.imread(filename=r"cat4.png")

height, width = img.shape[0], img.shape[1]

for i in range(height):

    for j in range(width):

        if img[i, j].sum() > 0:

            img[i, j] = 0

        else:

            img[i, j] = [255, 255, 255]

mask = img

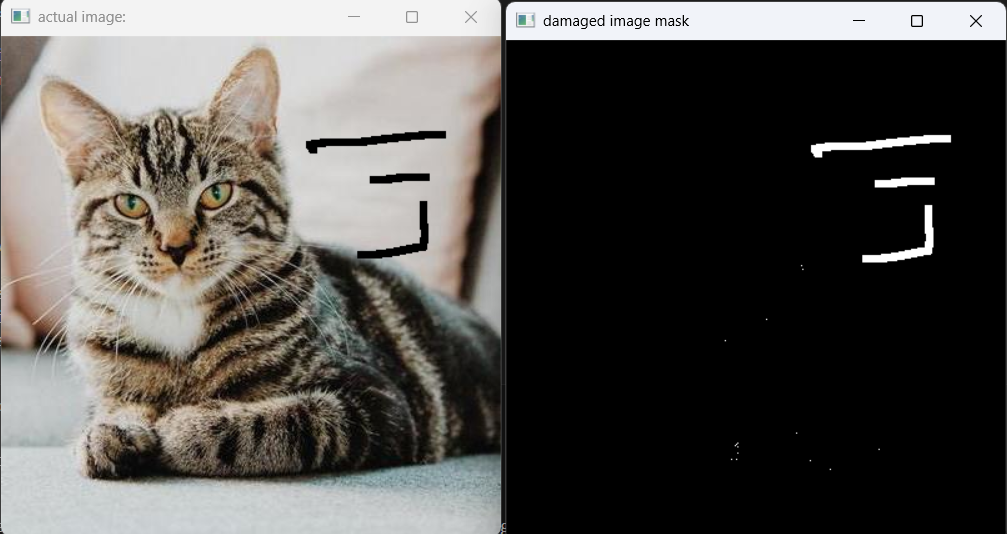
cv2.imwrite('mask.jpg', mask)

cv2.imshow("damaged image mask", mask)

cv2.waitKey(0)

cv2.destroyAllWindows()

OUTPUT:



**Background masking:**

import cv2

bg\_subtractor = cv2.createBackgroundSubtractorMOG2(history=100, varThreshold=40, detectShadows=True)

cap = cv2.VideoCapture('cat3.jpg')

while True:

    ret, frame = cap.read()

    if not ret:

        break

    frame = cv2.resize(frame, (640, 480))

    fg\_mask = bg\_subtractor.apply(frame)

    kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (3, 3))

    fg\_mask = cv2.morphologyEx(fg\_mask, cv2.MORPH\_OPEN, kernel)

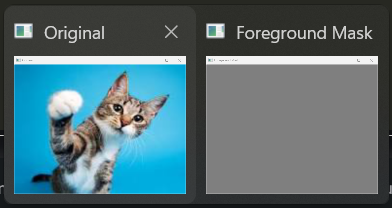
    cv2.imshow("Original", frame)

    cv2.imshow("Foreground Mask", fg\_mask)

   cap.release()

cv2.waitKey(0)

cv2.destroyAllWindows()



**IMAGE TRANSLATION:**

**CODE:**

import cv2

import numpy as np

image = cv2.imread('dog2.jpg')

height, width = image.shape[:2]

quarter\_height, quarter\_width = height / 4, width / 4

T = np.float32([[1, 0, quarter\_width], [0, 1, quarter\_height]])

img\_translation = cv2.warpAffine(image, T, (width, height))

cv2.imshow("Originalimage", image)

cv2.imshow('Translation', img\_translation)

cv2.waitKey()

cv2.destroyAllWindows()

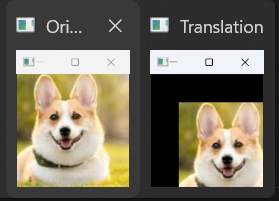


IMAGE PYRAMID:

Code:

import cv2

import matplotlib.pyplot as plt

img = cv2.imread('cat1.jpg')

layer = img.copy()

for i in range(4):

    plt.subplot(2, 2, i + 1)

    layer = cv2.pyrDown(layer)

    plt.imshow(layer)

    cv2.imshow("str(i)", layer)

    cv2.waitKey(0)

cv2.destroyAllWindows()



