HOW TO READ IMAGES  
--------------------  
import cv2 as c  
  
pic=c.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp") # for reading the image  
  
c.imshow('Dog',pic) # for displaying the image  
c.waitKey(0) #waits for infinte loop  
  
=================================  
HOW TO READ VIDEOS  
-------------------  
import cv2 as c  
  
Vid=c.VideoCapture(path)  
  
while True:  
 isTrue,frame=Vid.read()  
  
 c.imshow('video',frame)  
 if c.waitKey(20) & 0xFF==ord('d'):  
 break  
  
Vid.release()  
c.destroyAllwindows()  
  
=======================  
RESCALING(RESIZE)  
--------------  
\*rescaling: a video is absolutely imperative as it takes a lot of time for a large video  
\*Rescaling means resize its height and width to a particular dimensions (to a smaller dimensions)  
  
rescale for video:  
-----------------  
import cv2 as c  
def rescale(frame,scale=0.75):  
 width=int(frame.shape[1]\*scale)  
 height=int(frame.shape[0]\*scale)  
 dimensions=(width,height)  
 return c.resize(frame,dimensions,interpolation=c.INTER\_AREA)  
Vid=c.VideoCapture(path)  
while True:  
 isTrue,frame=Vid.read()  
  
 rescaleframe=rescale(frame)  
 if c.waitKey(20) & 0xFF==ord('d'):  
 break  
Vid.release()  
c.destroyAllwindows  
  
rescale for image:  
-----------------  
import cv2 as cv  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
  
cv.imshow('DOG',pic)  
cv.waitKey(0)  
  
def resize(frame,scale=.50):  
 width=int(frame.shape[1]\*scale)  
 height=int(frame.shape[0]\*scale)  
 dimensions=(width,height)  
 return cv.resize(frame,dimensions,interpolation=cv.INTER\_AREA)  
  
  
resized=resize(pic)  
cv.imshow('DOG2',resized)  
cv.waitKey(0)  
==================  
CREATING A BLANK AND COLOR IMAGE  
--------------------------------  
\*creating a blank image(black)  
-----------------------------  
import cv2 as cv  
import numpy as np  
  
blank=np.zeros((500,500),dtype='uint8') #uint8 is image dtype  
cv.imshow('BLANK',blank)  
cv.waitKey(0)  
----------------------  
\*Creating a color image:  
------------------------  
import cv2 as cv  
import numpy as np  
  
blank=np.zeros((500,500,3),dtype='uint8') #uint8 is image dtype, the third arugument in shape(500,500,3) refers to the colour  
blank[:]=0,255,0 # here we are creating a green color and by using the slice operator we select the pixels([:)  
cv.imshow('BLANK',blank)  
cv.waitKey(0)  
\*writing text to blank:  
---------------------  
import cv2 as cv  
import numpy as np  
  
blank=np.zeros((500,500,3),dtype='uint8')  
cv.putText(blank,'Hello Good Morning',(100,200),cv.FONT\_HERSHEY\_TRIPLEX,1.0,(0,0,255),2)  
cv.imshow('text',blank)  
cv.waitKey(0)  
=====================  
5 BASIC FUNCTIONS  
-----------------------  
\*CONVERTING INTO GRAY SCALE:  
--------------------------  
\*CONVERTS BGR TO GRAY SCALE TO SEE INTENSITY OF PIXELS  
  
import cv2 as cv  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
gray=cv.cvtColor(pic,cv.COLOR\_BGR2GRAY)# color code is cv.COLOR\_BGR2GRAY  
cv.imshow('gray',gray)  
cv.waitKey(0)  
------------------  
\*BLUR THE IMAGE  
------------------  
\*IT REMOVES SOME OF THE NOISE  
  
import cv2 as cv  
  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
BLUR=cv.GaussianBlur(pic,(3,3),cv.BORDER\_DEFAULT)# (3,3) is kernel size to increase the blur  
  
cv.imshow('Blur',BLUR)  
cv.waitKey(0)  
------------------  
\*EDGE CASCADE:(EDGE DETECTION)  
----------------  
\*VARIOUS WAYS TO DO THAT ONE WAY IS CANNY  
\*blur image is taken for edge detection and processed and being put into the dilated  
  
import cv2 as cv  
  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
canny=cv.Canny(pic,125,125,cv.BORDER\_DEFAULT)#2 threshold values is set  
  
cv.imshow('edge dection',canny)  
cv.waitKey(0)  
-----------------------  
\*DILATION:  
------------------------  
\* canny edges is processed to dilate image  
  
import cv2 as cv  
  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
canny=cv.Canny(pic,200,200)  
dilate=cv.dilate(canny,(3,5),iterations=3)  
cv.imshow('dilATED',dilate)  
cv.waitKey(0)  
----------------------  
\*ERODE  
--------------------  
\*is to restore structuring element(canny edges)  
\*dilated img is passed  
  
import cv2 as cv  
  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
canny=cv.Canny(pic,200,200)  
dilate=cv.dilate(canny,(3,5),iterations=3)  
erode=cv.erode(dilate,(3,5),iterations=3)  
cv.imshow('ERODE',erode)  
cv.waitKey(0)  
-------------------  
\*RESIZE:  
----------------  
\* to resize the width and the height  
  
import cv2 as cv  
  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
resized=cv.resize(pic,(300,200))  
cv.imshow('resize',resized)  
cv.waitKey(0)  
------------------------  
\*CROPPING:  
----------------  
import cv2 as cv  
  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
crop=pic[30:234,200:400]  
  
cv.imshow('cropping',crop)  
cv.waitKey(0)  
---------------------------  
TRANSFORMATION:  
------------------------  
\*TRANSALATION:  
-----------------  
\*moving the picture frame  
  
import cv2 as cv  
import numpy as np  
  
def translate(pic,x,y):  
 transmatrix=np.float32([[1,0,x],[0,1,y]])#creation of matrix  
 dimension=(pic.shape[1],pic.shape[0])  
 return cv.warpAffine(pic,transmatrix,dimension)  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
  
trans=translate(pic,400,200)# -x - left,x-right,-y-up,y-down  
cv.imshow('translation',trans)  
cv.waitKey(0)  
------------------  
\*ROTATION  
--------------  
import cv2 as cv  
import numpy as np  
  
def rotate(pic,angle,rotpoint=None):  
 (height,width)=pic.shape[:2]  
 if rotpoint==None:  
 rotpoint=(width//2,height//2)  
 rotmat=cv.getRotationMatrix2D(rotpoint,angle,1.0)  
 dimensions=(width,height)  
 return cv.warpAffine(pic,rotmat,dimensions)  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
  
rot=rotate(pic,45)  
cv.imshow('rotation',rot)  
cv.waitKey(0)  
------------------------  
\*RESIZE:  
---------------  
import cv2 as cv  
import numpy as np  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
  
resize=cv.resize(pic,(400,200),interpolation=cv.INTER\_CUBIC)  
cv.imshow('resize',resize)  
cv.waitKey(0)  
resize=cv.resize(pic,(400,200),interpolation=cv.INTER\_AREA)  
cv.imshow('resize',resize)  
cv.waitKey(0)  
------------------------------  
\*FLIP:  
--------------------------  
import cv2 as cv  
import numpy as np  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
  
  
flip=cv.flip(pic,-1)  
cv.imshow('flip',flip)  
cv.waitKey(0)  
===========================  
\*CONTOUR:  
-----------------------  
\* used in shape analysis,object detection  
\* not same as edges  
  
import cv2 as cv  
import numpy as np  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
pic=cv.resize(pic,(500,700))  
gray=cv.cvtColor(pic,cv.COLOR\_BGR2GRAY)  
cv.imshow('pic',gray)  
blur=cv.GaussianBlur(gray,(5,5),cv.BORDER\_DEFAULT)  
cv.imshow('blur',blur)  
canny=cv.Canny(blur,150,150,cv.BORDER\_DEFAULT)  
cv.imshow('edge',canny)  
  
contours,hierarchies=cv.findContours(canny,cv.RETR\_LIST,cv.CHAIN\_APPROX\_NONE)  
print(f'{(len(contours))} contours are there.')  
  
  
## ANOTHER WAY TO FIND THE CONTOUR  
  
pic=cv.imread('D:\OneDrive - LTTS\Desktop\golden retiver.webp')  
gray=cv.cvtColor(pic,cv.COLOR\_BGR2GRAY)  
ret,thresh=cv.threshold(gray,125,400,cv.THRESH\_BINARY)  
cv.imshow('binary',thresh)  
print(f'{len(thresh)}')  
cv.waitKey(0)  
-------------------------------------------  
\*\*DRAWING CONTOUR TO BLANK IMG  
  
import cv2 as cv  
import numpy as np  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
pic=cv.resize(pic,(500,700)) #resizeing the frame  
gray=cv.cvtColor(pic,cv.COLOR\_BGR2GRAY)  
cv.imshow('pic',gray)# converting into gray img  
blur=cv.GaussianBlur(gray,(5,5),cv.BORDER\_DEFAULT)  
cv.imshow('blur',blur)# blurring the img  
canny=cv.Canny(blur,150,150,cv.BORDER\_DEFAULT)  
cv.imshow('edge',canny)#finding the dege detection  
contours,hierarchies=cv.findContours(canny,cv.RETR\_LIST,cv.CHAIN\_APPROX\_NONE)  
blank=np.zeros(pic.shape,dtype='uint8')#creating a blank for drawing the contours  
  
  
cv.drawContours(blank,contours,-1,(0,255,0),2)  
cv.imshow('copy contour image',blank)# drawing the contours on the blank img  
cv.waitKey(0)  
==================================  
\*COLOR SPACING  
--------------------  
\*using matplotlib returns a bgr into rgb(its inversion)  
\*to get the orginal img we should convert bgr into rgb and plot it in matplotlib  
  
import cv2 as cv  
import matplotlib.pyplot as plt  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
pic=cv.resize(pic,(500,700)) #resizeing the frame  
cv.imshow('ORGINAL ',pic)  
gray=cv.cvtColor(pic,cv.COLOR\_BGR2GRAY)  
cv.imshow('pic',gray)# converting into gray img  
hsv=cv.cvtColor(pic,cv.COLOR\_BGR2HSV)  
cv.imshow('hsv',hsv)# converting pic to hue saturation value  
lab=cv.cvtColor(pic,cv.COLOR\_BGR2LAB)  
cv.imshow('LAB',lab)  
rgb=cv.cvtColor(pic,cv.COLOR\_BGR2RGB)#converting LAB  
cv.imshow('RGB',rgb)  
cv.waitKey(0)  
plt.imshow(rgb)  
plt.show()  
  
======================================================  
\*COLOR CHANNELS  
--------------------  
\*splitting the bgr into individual color components  
  
import cv2 as cv  
import matplotlib.pyplot as plt  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
pic=cv.resize(pic,(500,700)) #resizeing the frame  
cv.imshow('ORGINAL ',pic)  
  
blue,green,red=cv.split(pic)  
  
cv.imshow('blue',blue)  
cv.imshow('green',green)  
cv.imshow('red',red)  
cv.waitKey(0)  
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  
import cv2 as cv  
import matplotlib.pyplot as plt  
import numpy as np  
  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
pic=cv.resize(pic,(500,700)) #resizeing the frame  
cv.imshow('ORGINAL ',pic)  
  
blue,green,red=cv.split(pic)  
cv.imshow('blue',blue)  
cv.imshow('green',green)  
cv.imshow('red',red)  
  
blank=np.zeros(pic.shape[:2],dtype='uint8')  
cv.imshow('blank',blank)  
merge=cv.merge([blue,blank,c])  
cv.imshow('merging',merge)  
  
  
cv.waitKey(0)  
==============================================  
\*BLURRING TECHNIQUES:  
-------------------------  
\*this technoques reduces the noise from the image  
\*median removes salt and pepper noise compared to gaussian and average  
\*bilateral is the most effective one in advanced processing.it retains the edges of the pic while compared to others  
  
import cv2 as cv  
import matplotlib.pyplot as plt  
import numpy as np  
  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
pic=cv.resize(pic,(500,700)) #resizeing the frame  
cv.imshow('ORGINAL ',pic)  
  
#average blur  
avg=cv.blur(pic,(5,5))  
cv.imshow('Average',avg)  
#gaussian  
gauss=cv.GaussianBlur(pic,(5,5),0)  
cv.imshow('Gaussian blur',gauss)  
#Median blur  
median=cv.medianBlur(pic,5)#it takes only one values for kernel  
cv.imshow('median',median)  
#bilateral blur  
bi=cv.bilateralFilter(pic,5,125,125)#diameter,sigma color,sigma space  
cv.imshow('BILATERAL',bi)  
  
cv.waitKey(0)  
====================================  
\*BITWISE OPERATORS  
-----------------------  
  
import cv2 as cv  
import numpy as np  
  
blank=np.zeros((400,400),dtype='uint8')  
  
rectangle=cv.rectangle(blank.copy(),(30,30),(370,370),255,-1)  
circle=cv.circle(blank.copy(),(200,200),200,255,-1)  
cv.imshow('rectangel',rectangle)  
cv.imshow('cirle',circle)  
#intersection regions  
bit\_AND=cv.bitwise\_and(rectangle,circle)  
cv.imshow('bitwise',bit\_AND)  
#non intersectiong regions and intersecting  
bit\_or=cv.bitwise\_or(rectangle,circle)  
cv.imshow('OR',bit\_or)  
#non intersecting regions  
bit\_Exor=cv.bitwise\_xor(rectangle,circle)  
cv.imshow('exor',bit\_Exor)  
#return a inverted(white->black and black->white)  
bit\_not=cv.bitwise\_not(rectangle)  
cv.imshow('not',bit\_not)  
x=bit\_AND-bit\_Exor  
cv.imshow('-',x)  
cv.waitKey(0)  
=================================  
\*MASKING  
------------------------------------  
\*allows to focus on certain parts(ie.peoples faces)  
  
import cv2 as cv  
import numpy as np  
  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
cv.imshow('original',pic)  
blank=np.zeros(pic.shape[:2],dtype='uint8')  
cv.imshow('blank',blank)  
mask=cv.circle(blank,(pic.shape[1]//2+40,pic.shape[0]//2-90),300,255,-1)  
cv.imshow('mask',mask)  
  
maked=cv.bitwise\_and(pic,pic,mask=mask)  
cv.imshow('masked pic',maked)  
cv.waitKey(0)  
===========================================  
\*HISTOGRAM(GRAY PIC)  
--------------------------  
\*allow to visulaize distributon the pixels of any image  
  
import cv2 as cv  
import numpy as np  
import matplotlib.pyplot as plt  
  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
resize=cv.resize(pic,(600,700))  
cv.imshow('original',resize)  
gray=cv.cvtColor(resize,cv.COLOR\_BGR2GRAY)  
cv.imshow('gray',gray)  
  
gray\_hist=cv.calcHist([gray],[0],None,[256],[0,256])  
  
plt.figure()  
plt.title('gray-histo')  
plt.xlabel('Bins')  
plt.ylabel('no.of pixels')  
plt.plot(gray\_hist)  
plt.xlim([0,256])  
plt.show()  
cv.waitKey(0)  
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  
\*With masking  
import cv2 as cv  
import numpy as np  
import matplotlib.pyplot as plt  
  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
cv.imshow('original',pic)  
blank=np.zeros(pic.shape[:2],dtype='uint8')  
cv.imshow('blank',blank)  
gray=cv.cvtColor(pic,cv.COLOR\_BGR2GRAY)  
mask=cv.circle(blank,(pic.shape[1]//2+40,pic.shape[0]//2-90),300,255,-1)  
cv.imshow('mask',mask)  
  
maked=cv.bitwise\_and(gray,gray,mask=mask)  
cv.imshow('masked pic',maked)  
  
gray\_hist=cv.calcHist([gray],[0],mask,[256],[0,256])  
  
plt.figure()  
plt.title('gray-histo')  
plt.xlabel('Bins')  
plt.ylabel('no.of pixels')  
plt.plot(gray\_hist)  
plt.xlim([0,256])  
plt.show()  
cv.waitKey(0)  
------------------------------------  
\*HISTOGRAM for RGB:  
-----------------------------  
  
import cv2 as cv  
import numpy as np  
import matplotlib.pyplot as plt  
  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
cv.imshow('original',pic)  
plt.figure()  
plt.title('color-Histo')  
plt.xlabel('Bins')  
plt.ylabel('no.of pixels')  
colors=('b','g','r')  
for i,col in enumerate(colors):  
 hist = cv.calcHist([pic], [i],None, [256], [0, 256])  
 plt.plot(hist,color=col)  
 plt.xlim([0, 256])  
  
plt.show()  
cv.waitKey(0)  
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  
\*FOR MASKED  
--------------------------------  
  
import cv2 as cv  
import numpy as np  
import matplotlib.pyplot as plt  
  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
cv.imshow('original',pic)  
blank=np.zeros(pic.shape[:2],dtype='uint8')  
#cv.imshow('blank',blank)  
gray=cv.cvtColor(pic,cv.COLOR\_BGR2GRAY)  
mask=cv.circle(blank,(pic.shape[1]//2+40,pic.shape[0]//2-90),300,255,-1)  
#cv.imshow('mask',mask)  
maked=cv.bitwise\_and(gray,gray,mask=mask)  
cv.imshow('masked pic',maked)  
  
plt.figure()  
plt.title('gray-histo')  
plt.xlabel('Bins')  
plt.ylabel('no.of pixels')  
colors=('b','g','r')  
for i,col in enumerate(colors):  
 hist = cv.calcHist([pic], [i],mask, [256], [0, 256])  
 plt.plot(hist,color=col)  
 plt.xlim([0, 256])  
  
plt.show()  
cv.waitKey(0)  
========================================  
\*THRESHOLDING  
---------------------------  
\*image converted into binary format either black(0) or white(255)  
\*types-simple and adaptive  
  
  
import cv2 as cv  
import numpy as np  
import matplotlib.pyplot as plt  
  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
cv.imshow('original',pic)  
gray=cv.cvtColor(pic,cv.COLOR\_BGR2GRAY)  
  
#simple thresholding  
threshold,thresh=cv.threshold(gray,150,255,cv.THRESH\_BINARY)#150-threshold value,255-white color, mode  
cv.imshow('threshold',thresh)  
  
#INverse  
threshold,thresh\_inverse=cv.threshold(gray,150,255,cv.THRESH\_BINARY\_INV)  
cv.imshow('thresholdinverse',thresh\_inverse)  
  
##ADAPATIVE THRESHOLD  
  
adaptive=cv.adaptiveThreshold(gray,255,cv.ADAPTIVE\_THRESH\_MEAN\_C,cv.THRESH\_BINARY,11,5)  
#computes mean to set a threshold(11(blocksize)-kenerl size:to compute mean),c=5 to subtract from mean  
cv.imshow('adaptive thershold',adaptive)  
  
cv.waitKey(0)  
==============================================  
\*EDGE DETECTION:  
------------------------------  
\*types-laplacian and sobel  
\*canny edge detector uses multistage process so it gives better results  
  
import cv2 as cv  
import numpy as np  
import matplotlib.pyplot as plt  
  
#pic=cv.imread("D:\OneDrive - LTTS\Desktop\golden retiver.webp")  
pic=cv.imread("D:\OneDrive - LTTS\Desktop\cristiano-ronaldo-manchester.jpg")  
cv.imshow('original',pic)  
gray=cv.cvtColor(pic,cv.COLOR\_BGR2GRAY)  
#SOBEL  
lap=cv.Laplacian(gray,cv.CV\_64F) # cv\_64 is data depth  
lap=np.uint8(np.absolute(lap)) # here we use numpy to convert all values to postive values and to img datatye(uint8)  
cv.imshow('Laplaican',lap)  
#SOBEL  
sobel\_x=cv.Sobel(gray,cv.CV\_64F,1,0)  
sobel\_y=cv.Sobel(gray,cv.CV\_64F,0,1)  
combine=cv.bitwise\_and(sobel\_y,sobel\_x)  
cv.imshow('Sobel X',sobel\_x)  
cv.imshow('Sobel Y',sobel\_y)  
cv.imshow('Combine',combine)  
  
cv.waitKey(0)  
==============================================  
HAAR CASCADE(FACE DETECTION:  
---------------------------------  
  
===========================  
OPenCV BUILT IN FACE DETECTOR  
-----------------------------------