Basic commands

22 May 2022 01:56 PM

Import cv -> importing the computer vision package

To read an image -> cv2.imread("image with path")
Img = cv2.imread("users/adi.jpg")

To display an image - > cv2.imshow("Window name", image variable name)
Cv2.imshow("Output Image", img)

To make the output window stay infinite seconds -> cv2.waitkey(0)

To read and display a video ->
Import cv2

Cap = cv2.VideoCapture("path and file name with extension")
While True:

Success, img = cap.read()	#success is a boolean variable

Cv2.imshow("Video", img)
If cv2.waitKey(1) & 0xFF == ord('q'):
break

this procedure is followed since a video is typically a series of images running together. If loop for pressing q to exit

To use a webcam ->

Import cv2

Cap = cv2.VideoCapture(0) #0 is the id number of the webcam

Cap.set(3,640)	The first attribute points to the dimension, 3 is the id for width and the second attribute is the actual width value.
Cap.set(4,480)	The first attribute points to the dimension, 4 is the id for height and the second attribute is the actual height value.
Cap.set(10,100)	10 is the id for the brightness and the second attribute is the actual value.

#Set() method is used to specify the dimensions.
While True:

Success, img = cap.read()	#success is a boolean variable

Cv2.imshow("Video", img)
If cv2.waitKey(1) & 0xFF == ord('q'):
break

Redundant Methods to know

22 May 2022 03:49 PM

To change an image from color to grayscale -> cv2.cvtColor(source, COLOR_BGR2GRAY)

To blur an image -> cv2.GaussianBlur(source, (kernal_size)[always must be odd], sigmax)

To detect the edges and just display them -> cv2.Canny(source, Threshold1,Threshold2). The threshold values are altered as per the use case

sometimes we are detecting an edge, but because there is a gap, or because it's not joined properly, it does not detect it as a proper line. So what we can do is we can increase the thickness of our edge. So in order to do that, we will use image dilation -> cv2.dialate()
(eg).

```
importcv2
importnumpyasnp

img=cv2.imread("assets/hhh.jpg")
kernel=np.ones((5,5),np.uint8) #kernel is defined so it can be passed as a paramter to the cv2.dilate method.

imgGray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
imgBlur=cv2.GaussianBlur(imgGray,(7,7),0)
imgCanny=cv2.Canny(imgGray,150,200)
imgDilate=cv2.dilate(imgCanny,kernel,iterations=1)
#cv2.imshow("GrayedOutput",imgGray)
#cv2.imshow("BlurredOutput",imgBlur)
cv2.imshow("CannyOutput",imgCanny)
cv2.imshow("DilatedOutput",imgDilate)

cv2.waitKey(0)
```

The opposite of dilation is erode function, which makes the lines less deeper

Resizing and cropping

22 May 2022 05:14 PM

In general, the positive x axis is towards the east and the positive y axis is towards the north, but in opencv, the positive x axis is the same, but the positive y axis extends towards the south and the corresponding points are plotted

To resize an image ->
importcv2

img=cv2.imread("assets/adi.jpg")
print(img.shape) #(3429, 2289, 3) - this is the output, where the first value is the height, second is the width, third is
The rgb color value

imgResized=cv2.resize(img,(500,600)) #while setting the parameter values, the first is the width value, second is the
height value

cv2.imshow("imgResized",imgResized)
cv2.waitKey(0)

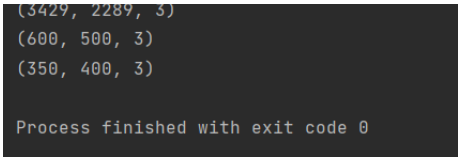
```
import cv2

img = cv2.imread("assets/adi.jpg")
print(img.shape)

imgResized = cv2.resize(img, (500, 600))
print(imgResized.shape)

cv2.imshow("imgResized", imgResized)
cv2.waitKey(0)
```

```
main ×
C:\Users\ravia\PycharmProjects\pythonProje
(3429, 2289, 3)
(600, 500, 3)
```

To crop an image ->
Now image itself is just a matrix or an array of pixels. So what we can do is we can deal it in terms of an array or a matrix. So what we can do is we can write image corrupt is equals to image, so this is our main image that we want to crop but now we don't need an opencv function. We can just use the matrix functionality. So we can say, we can define the starting point and the ending point for both our width and height. So for example, **now this is a little bit tricky because the height comes first and then the width but in the opencv function, the width came first and then the height.**

```
import cv2

img = cv2.imread("assets/adi.jpg")
print(img.shape)

imgResized = cv2.resize(img, (500, 600))
print(imgResized.shape)

imgCropped = img[250:600, 100:500]

#cv2.imshow("imgResized", imgResized)
cv2.imshow("img cropped", imgCropped)
print(imgCropped.shape)
cv2.waitKey(0)
```

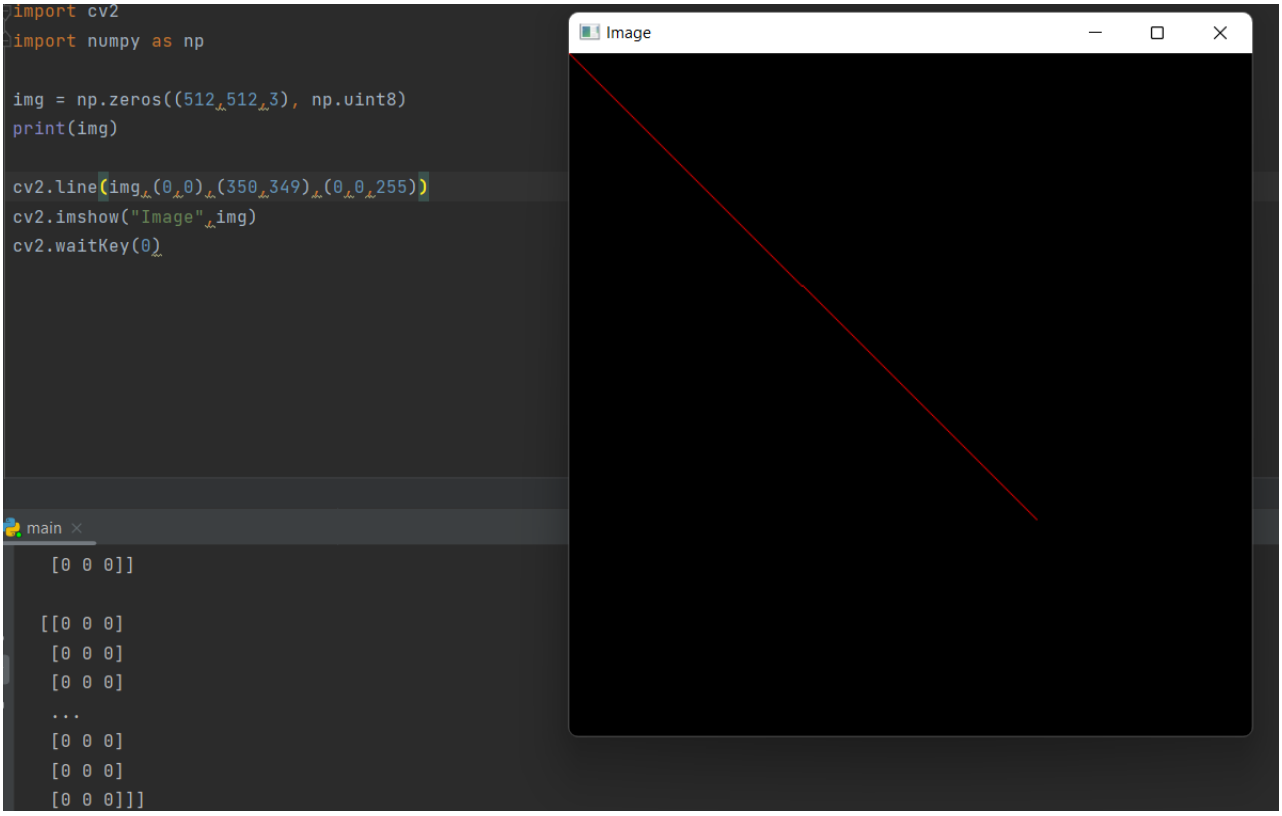
```
main ×
C:\Users\ravia\PycharmProjects\pythonProj
```


(3429, 2289, 3)
(600, 500, 3)
(350, 400, 3)
Process finished with exit code 0

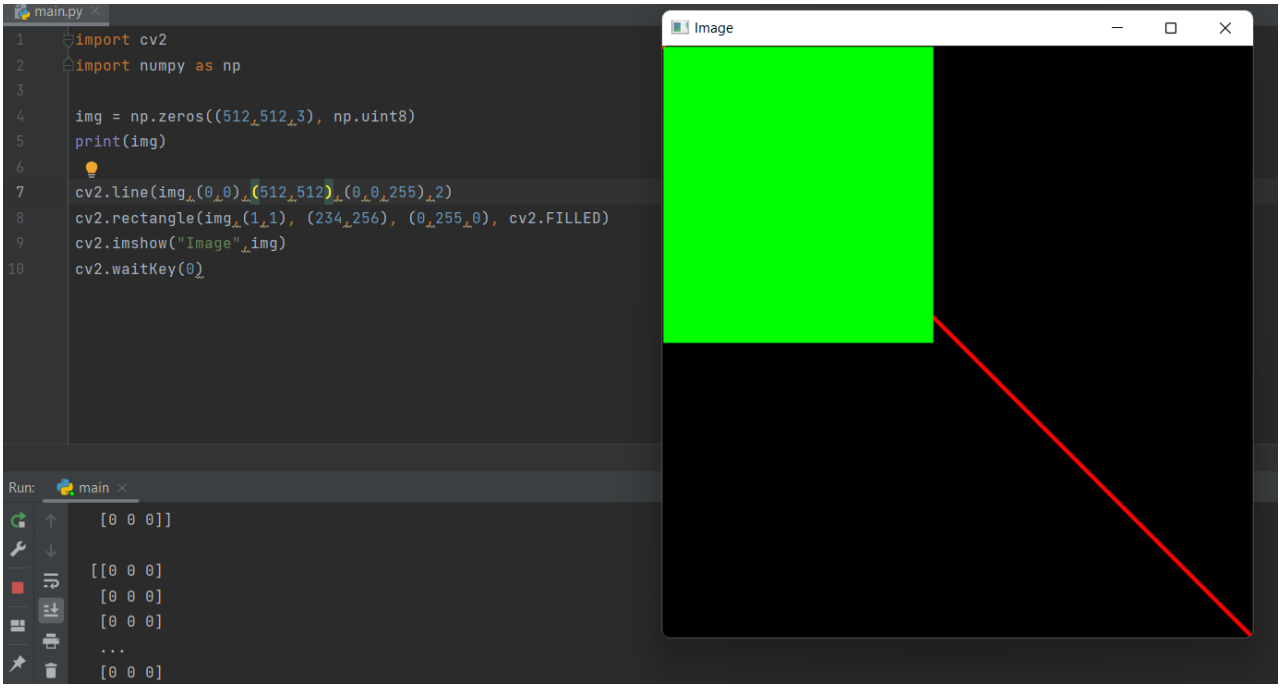
Shapes and texts

22 May 2022 05:42 PM

Line: syntax = cv2.line(source,point1,point2,color,thickness)

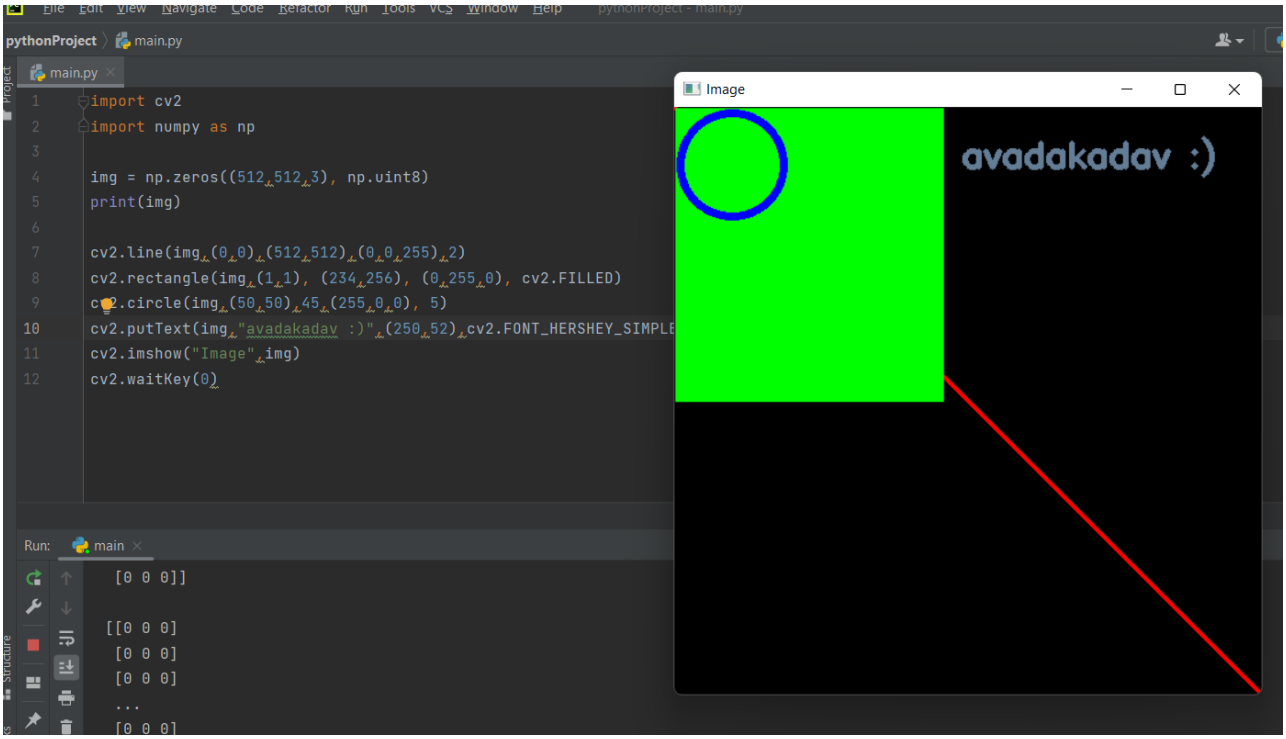


Rectangle: syntax = cv2.rectangle(source,point1,point2,color,thickness) --> to fill the rectangle, we can use cv2.FILLED in place of thickness



Circle: syntax = cv2.circle(source,center points, radius, color, thickness)

Add text to an image: syntax = cv2.putText(source,"text to be added", origin points, fontface, fontscale, color, thickness)



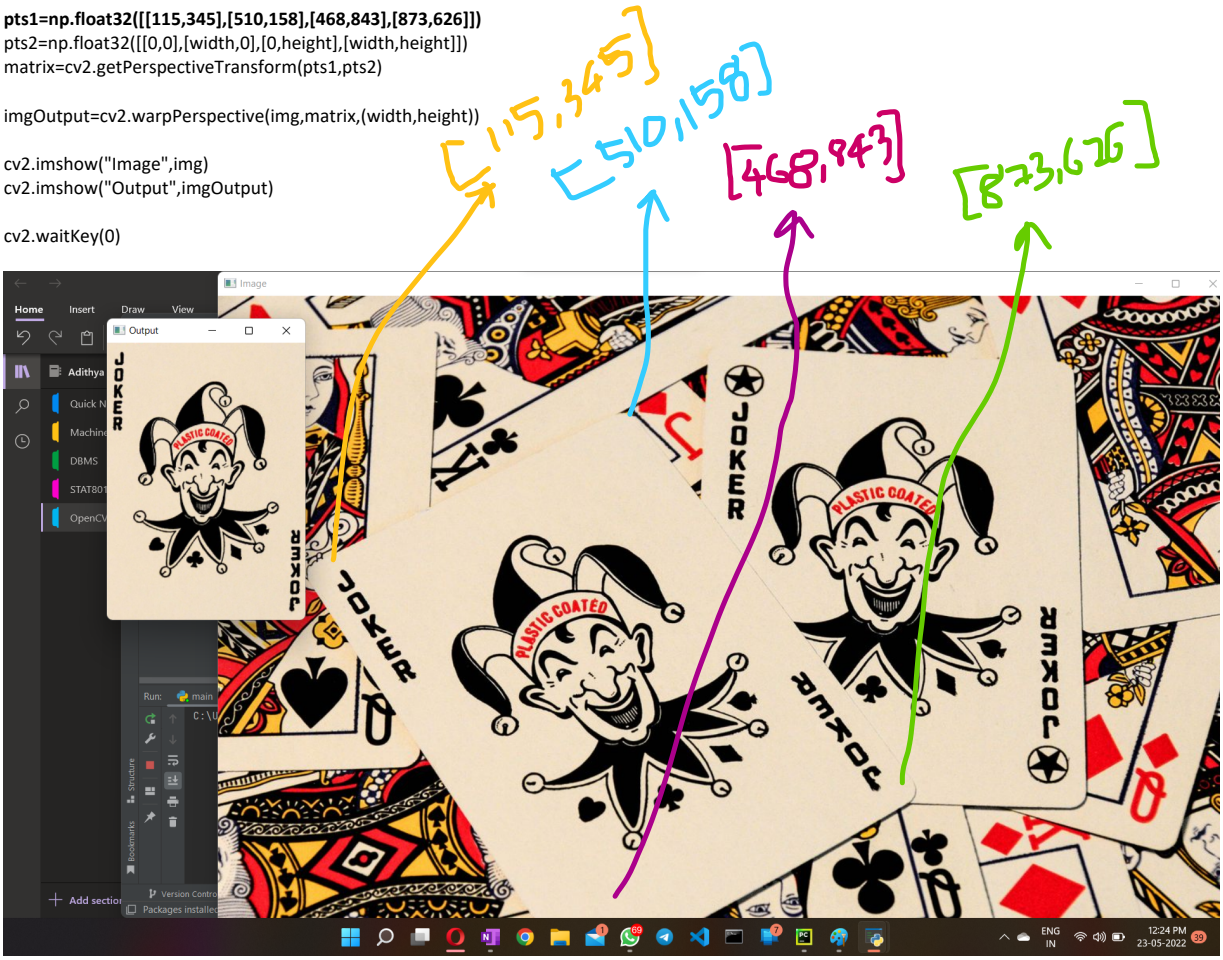
Warp perceptive

22 May 2022 06:51 PM

Warp perspective is used in situations where, a part of an image has to be cropped and the cropped image has to be given as output with a proper upright perspective. To do this, opencv has inbuilt methods like getPersperctiveTransform(point1,point2) and warpPerspective(source, matrix, (width,height)).

The point1 and point2 are the points where, point1 is the coordinates of the part of image for which the warp perspective has to be performed and point2 is the coordinates of the entire image.

```
importcv2
importnumpyasnp

img=cv2.imread("assets/cards.jpg")

width,height=250,350

pts1=np.float32([[115,345],[510,158],[468,843],[873,626]])
pts2=np.float32([[0,0],[width,0],[0,height],[width,height]])
matrix=cv2.getPerspectiveTransform(pts1,pts2)

imgOutput=cv2.warpPerspective(img,matrix,(width,height))

cv2.imshow("Image",img)
cv2.imshow("Output",imgOutput)

cv2.waitKey(0)
```

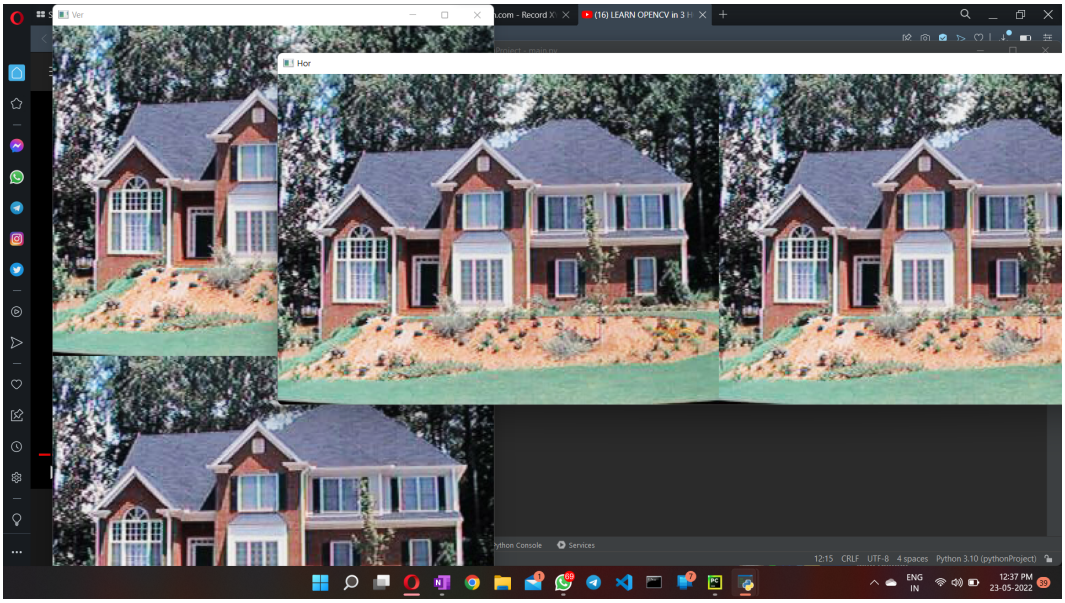


Joining Images

23 May 2022 12:28 PM

Joining images becomes necessary when we have too many images that are redundantly used. These commonly used images can be joined together by using the numpy functions np.hstack(), np.vstack().

```
importnumpyasnp

Import cv2
Import numpy as np

img=cv2.imread("assets/hhh.jpg")

imgHor=np.hstack((img,img))
imgVer=np.vstack((img,img))

cv2.imshow("Hor",imgHor)
cv2.imshow("Ver",imgVer)

cv2.waitKey(0)
```



The disadvantage that comes with this is that, when the images are not of same channel, that is, when one is greyscale and the other is rgb, this function wouldnt work. And another con is the image sizes cannot be resized, so when we stack it multiple times - the image would occupy the entire screen and sometimes would not be able to see the entire image at all. To avoid that the followingv procedure is followed

```
importcv2
importnumpyasnp

defstackImages(scale,imgArray):
rows=len(imgArray)
cols=len(imgArray[0])
rowsAvailable=isinstance(imgArray[0],list)
width=imgArray[0][0].shape[1]
height=imgArray[0][0].shape[0]
ifrowsAvailable:
forxinrange(0,rows):
foryinrange(0,cols):
ifimgArray[x][y].shape[:2]==imgArray[0][0].shape[:2]:
imgArray[x][y]=cv2.resize(imgArray[x][y],(0,0),None,scale,scale)
else:
imgArray[x][y]=cv2.resize(imgArray[x][y],(imgArray[0][0].shape[1],imgArray[0][0].shape[0]),
None,scale,scale)
iflen(imgArray[x][y].shape)==2:imgArray[x][y]=cv2.cvtColor(imgArray[x][y],cv2.COLOR_GRAY2BGR)
imageBlank=np.zeros((height,width,3),np.uint8)
hor=[imageBlank]*rows
hor_con=[imageBlank]*rows
forxinrange(0,rows):
hor[x]=np.hstack(imgArray[x])
ver=np.vstack(hor)
else:
forxinrange(0,rows):
ifimgArray[x].shape[:2]==imgArray[0].shape[:2]:
imgArray[x]=cv2.resize(imgArray[x],(0,0),None,scale,scale)
else:
```

```
imgArray[x]=cv2.resize(imgArray[x],(imgArray[0].shape[1],imgArray[0].shape[0]),None,scale,scale)
iflen(imgArray[x].shape)==2:imgArray[x]=cv2.cvtColor(imgArray[x],cv2.COLOR_GRAY2BGR)
hor=np.hstack(imgArray)
ver=hor
Return ver

img=cv2.imread("assets/hhh.jpg")
imgGrey=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
imgStack=stackImages([img,img,imgGrey],1)

cv2.imshow("StackedImagesUsingFunction",imgStack)

cv2.waitKey(0)
```