INDEX

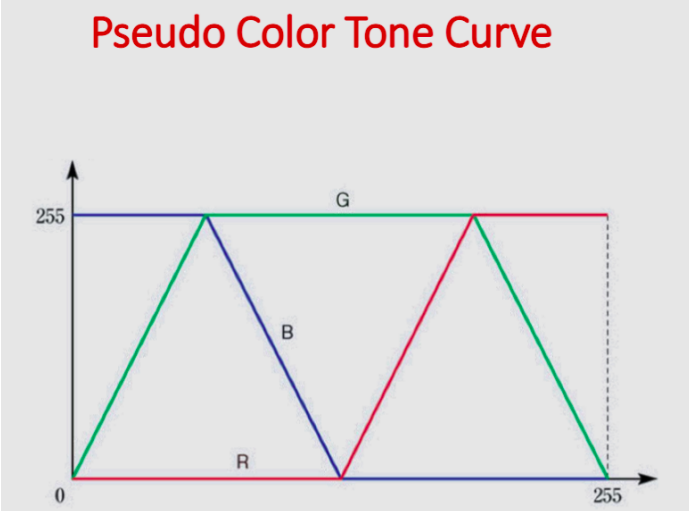
1. Method used
2. Output Images screenshot
3. Readme screenshot
4. Code screenshot

METHOD USED:

Steps:

1. Read the input image from the filename
2. Convert to grayscale for finding the
    a. Brightest pixel
    b. Lightest pixel
    c. location(centroid) of the brightest pixel
3. Making the lookup table
    a. As we want the range to be in pixel_min to pixel_max, we divide this range into 4, as any one of the RGB color functions changes in the 4 intervals. The 4 intervals are defined as below.
        i. $step = (pixelmax - pixelmin)/4$
        ii. Interval 1: $pixelmin\ to\ pixelmin + step$
        iii. Interval 2: $pixelmin + step\ to\ pixelmin + 2 * step$
        iv. Interval 3: $pixelmin + 2 * step\ to\ pixelmin + 3 * step$
        v. Interval 4: $pixelmin + 3 * step\ to\ pixelmax$

    For each value in the range, a specific RGB is assigned in the LUT.
4. Using the cv2.LUT function to get the pseudo color image
5. Saving the image
6. Displaying the image



The tone curve used(replaced 255 in the x axis with pixel_max, and 0 with pixel_min)

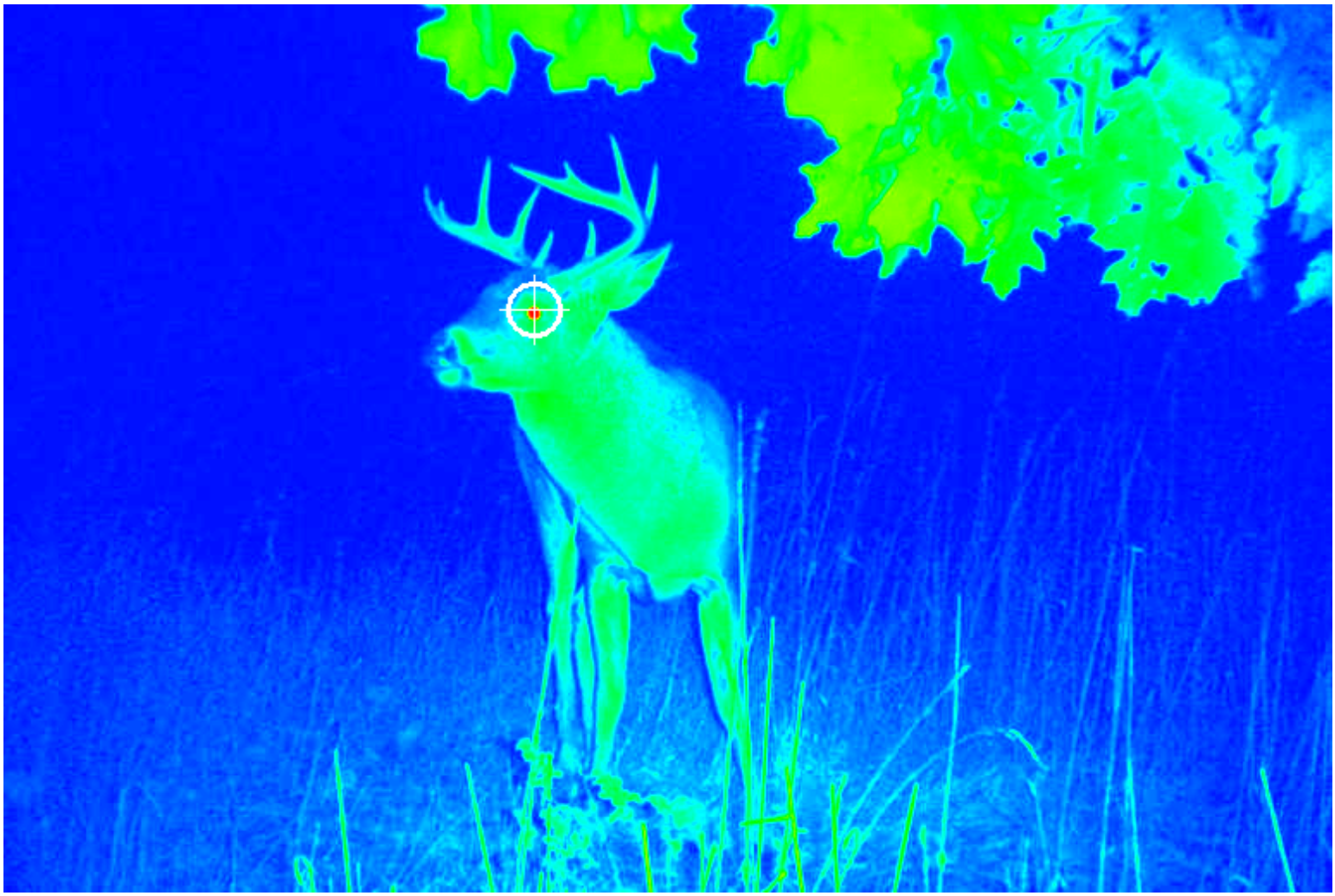
Code Screenshot

```
import cv2
import numpy as np
import os


#asking the reader for filename
filename = input("Enter the filename of the image : ")

#filename = "/home/akshay/Downloads/CV/ps-2-q/ps2-images/x-ray.png"
#reading file from file name
input_image = cv2.imread(filename)

#Displaying the input
cv2.imshow("input image",input_image)

#changing input to single channel for ease of operation of finding brightest pixel
input_gray = cv2.cvtColor(input_image,code=cv2.COLOR_BGR2GRAY)

#calculating maximum and minimum pixel values, and step length so the
#range(pixel_min,pixel_max) is divided into 4
pixel_max = np.amax(input_gray)
pixel_min = np.amin(input_gray)
step_length = int((pixel_max - pixel_min)/4)
m, n, _ = input_image.shape

#x will denote the sum of x coordinates with maximum gray value so as y, n
# denotes the coordinates with maximum gray values
x, y, number = 0,0,0
max_gray_val_list = []

#taking all the coordinates of maximum pixel values
for i in range(m):
    for j in range(n):
        if input_gray[i,j] == pixel_max:
            number += 1
            x += i
            y += j

#calculating centroid for finding the brightest pixel
centre_coordinates = (int(y/number),int(x/number))

#initializing look up table with 0 values
lut = np.zeros((256,1,3), dtype=np.uint8)

#iterating separately for 4 steps as either R,G,B changes the value in each step, so minimum is 4
#the range pixel_max-pixel_min is divided into 4, and appropriate values are given to RGB channels
#according to the graph discussed in the class
for i in range(pixel_min,pixel_min+step_length,1):
    lut[i][0][0] = 255
    lut[i][0][1] = (255/step_length)*(i - pixel_min)
    lut[i][0][2] = 0

for j in range(pixel_min + step_length , pixel_min + 2*step_length,1):
    lut[j][0][0] = 255 - (255/step_length)*(j - pixel_min - step_length)
    lut[j][0][1] = 255
    lut[j][0][2] = 0

for k in range(pixel_min + 2*step_length,pixel_min + 3*step_length,1):
    lut[k][0][0] = 0
    lut[k][0][1] = 255
    lut[k][0][2] = (255/step_length)*(k - pixel_min - 2*step_length)

for l in range(pixel_min + 3*step_length, pixel_max + 1,1):
    lut[l][0][0] = 0
    lut[l][0][1] = 255 - (255/(pixel_max - pixel_min -3*step_length))*(l - pixel_min - 3*step_length
    lut[l][0][2] = 255

#initialising output_image with 0
```

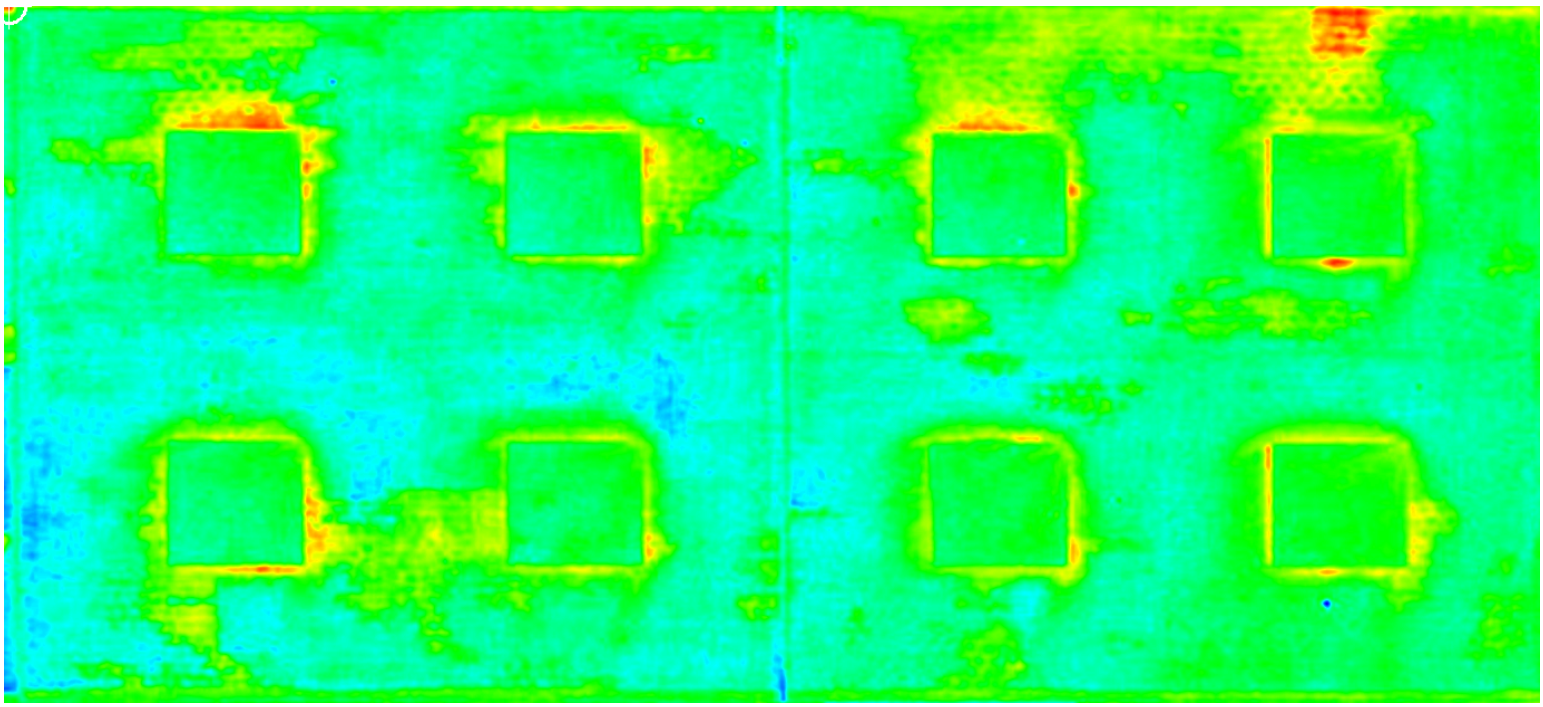
```
output_image = np.zeros_like(input_image)

#converting the input grayscale based on the lookup table we made
output_image = cv2.LUT(input_image,lut)

#defining start and end points for the cross(perpendicular lines) with 40 length
line_e = (centre_coordinates[0]+20,centre_coordinates[1])
line_w = (centre_coordinates[0]-20,centre_coordinates[1])
line_n = (centre_coordinates[0],centre_coordinates[1]+20)
line_s = (centre_coordinates[0],centre_coordinates[1]-20)

#drawing circle at the centroid
output_image = cv2.circle(output_image,centre_coordinates,15,(255,255,255),2)
output_image = cv2.line(output_image,line_w,line_e,(255,255,255),1)
output_image = cv2.line(output_image,line_s,line_n,(255,255,255),1)


#finding the name of the image
first_name = os.path.basename(filename)
first_name = first_name.split('.')[0]

#writing image to the path where this solution python file is stored
path = "/home/akshay/Downloads/CV/ps2-1/" + first_name + "-color.png"

writeStatus = cv2.imwrite(path, output_image)
if writeStatus is True:
    print("image written successfully")
else:
    print("image writing failed")

#displaying the output line
cv2.imshow("output image",output_image)
cv2.waitKey(0)
```

Output Images

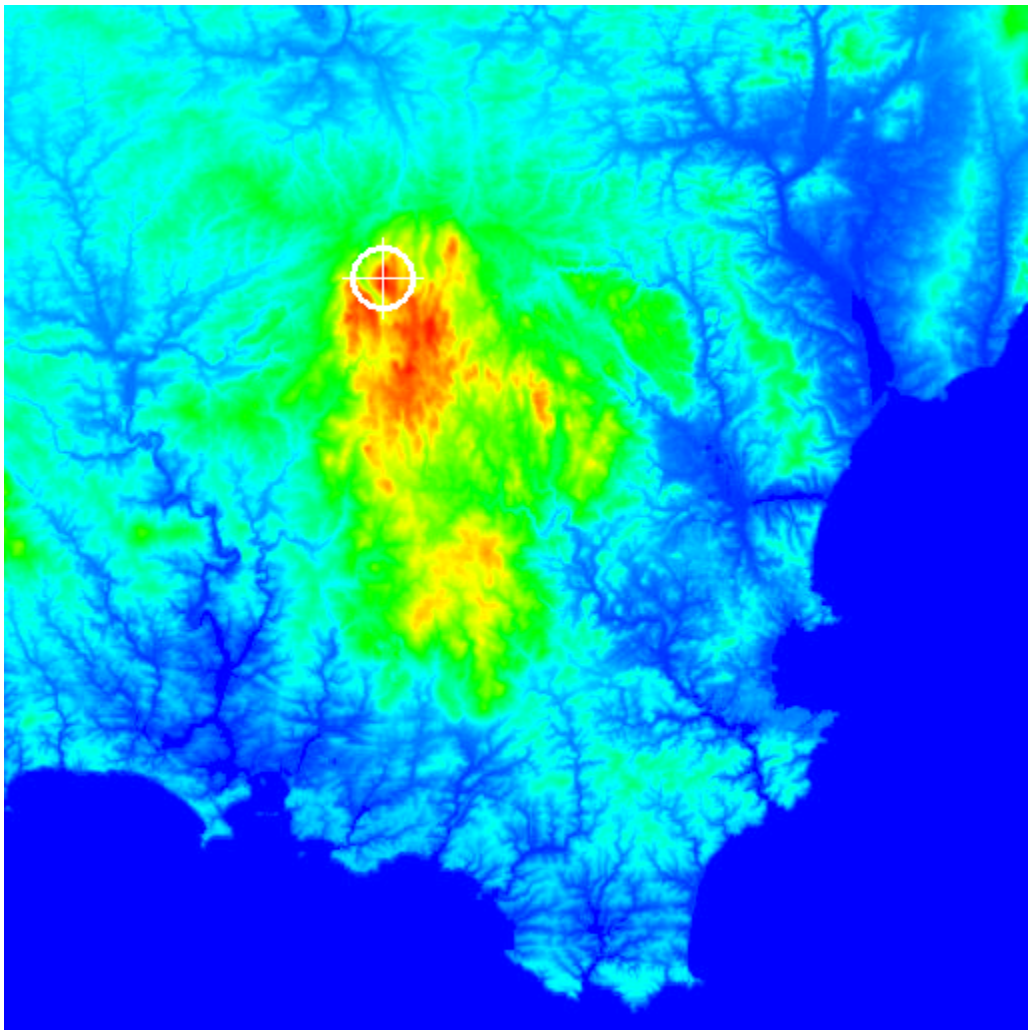
night-vision.png



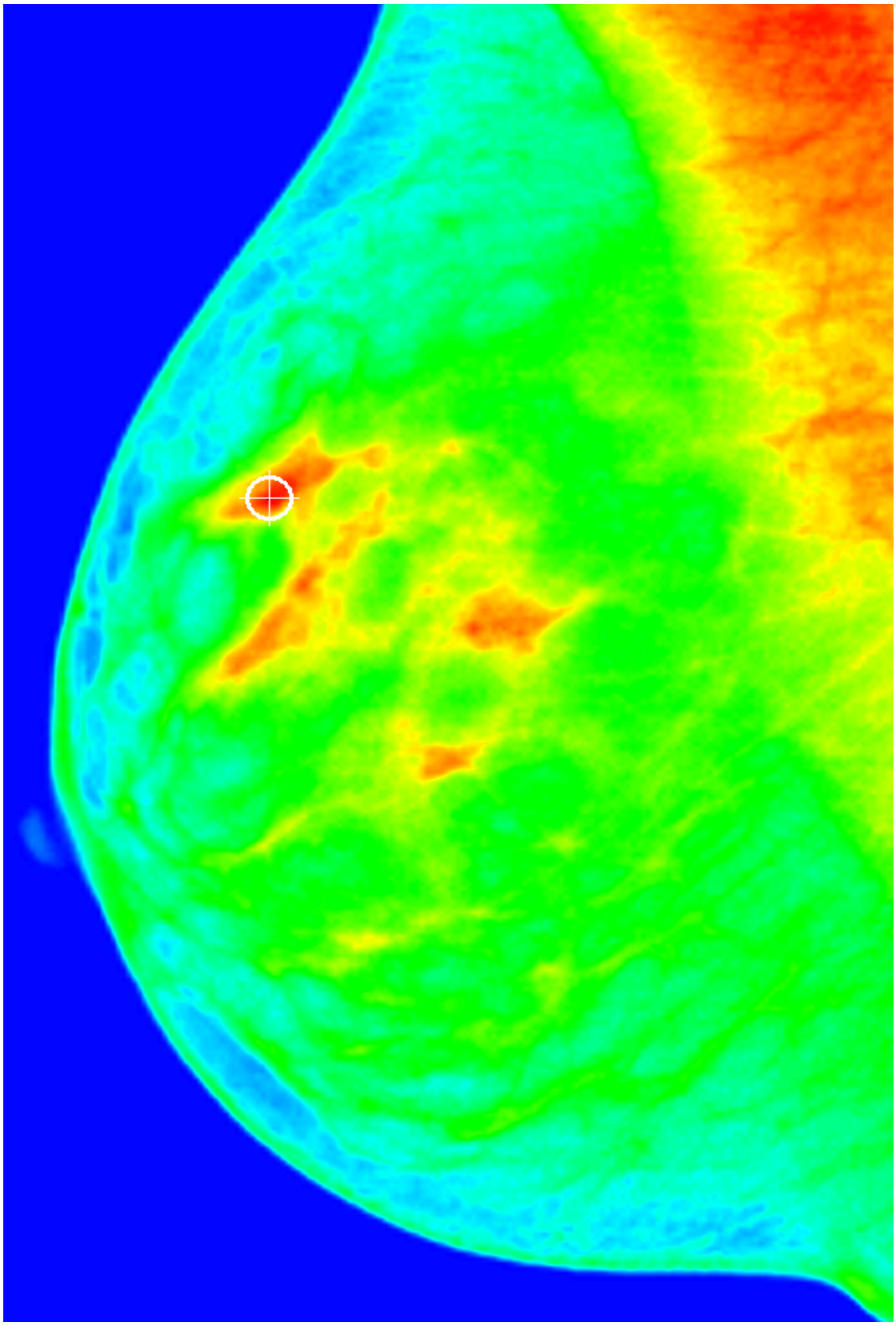
thermal-color.png



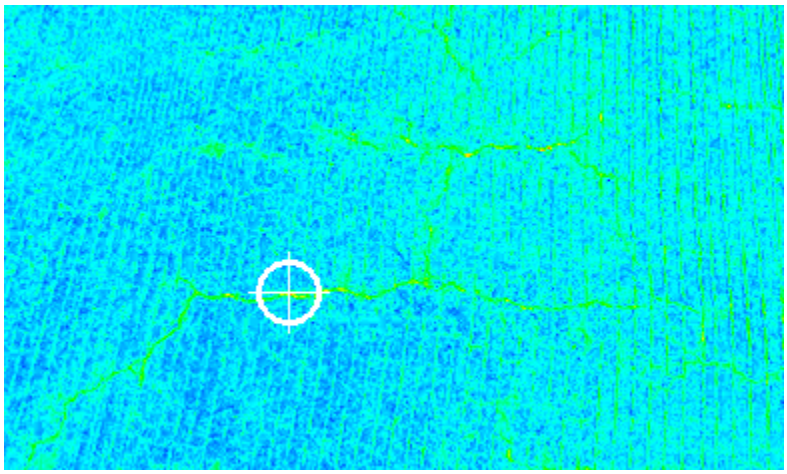
topography-color.png



x-ray-color.png



cracks-color.png



Readme Screenshot


```
Python version: Python 3.9.6
OpenCV Version: 4.5.2
Operating Systenm: Linux 20.02
IDE: Sublime text, run via terminal
Almost spend 6 hours for this problem
```