PS-5 REPORT
24-678

SUBMITTED BY: AKSHAY ANTONY
akshayan@andrew.cmu.edu

Pseudo Code:

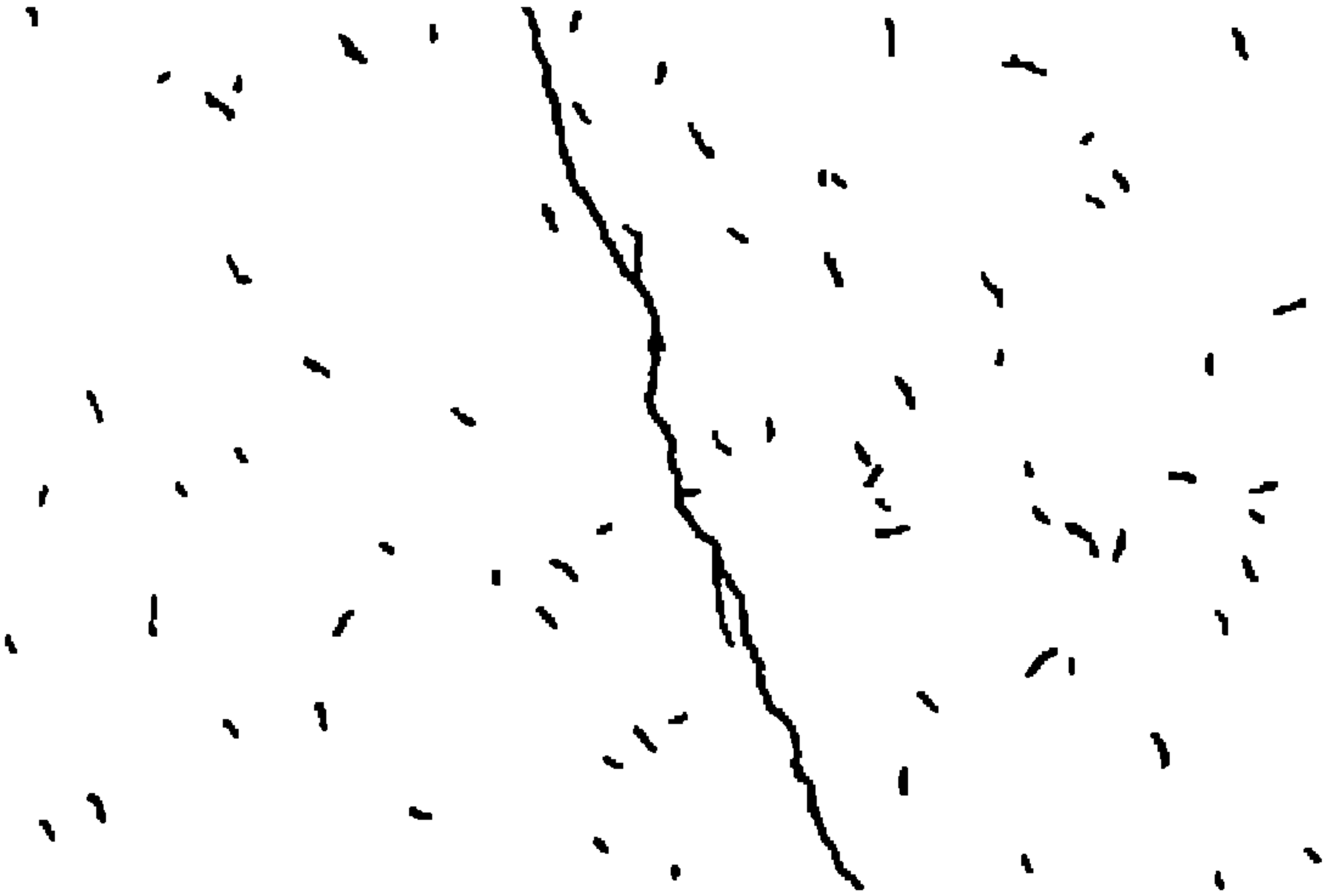
1. Input the image
2. Do the required number of erosions and dilations to obtain **wall-blobs.png**
3. Use cv2.findContour to find all the contours
4. Use random colors to draw around different blobs and save them as **wall-contours.png**
5. Threshold out required contours based on the **arc length, using cv2.arcLength** function
6. Draw the thresholded contours on a separate white screen
7. Thin the contour to find the central axis and save it as **wall-cracks.png**

Wall-1.png

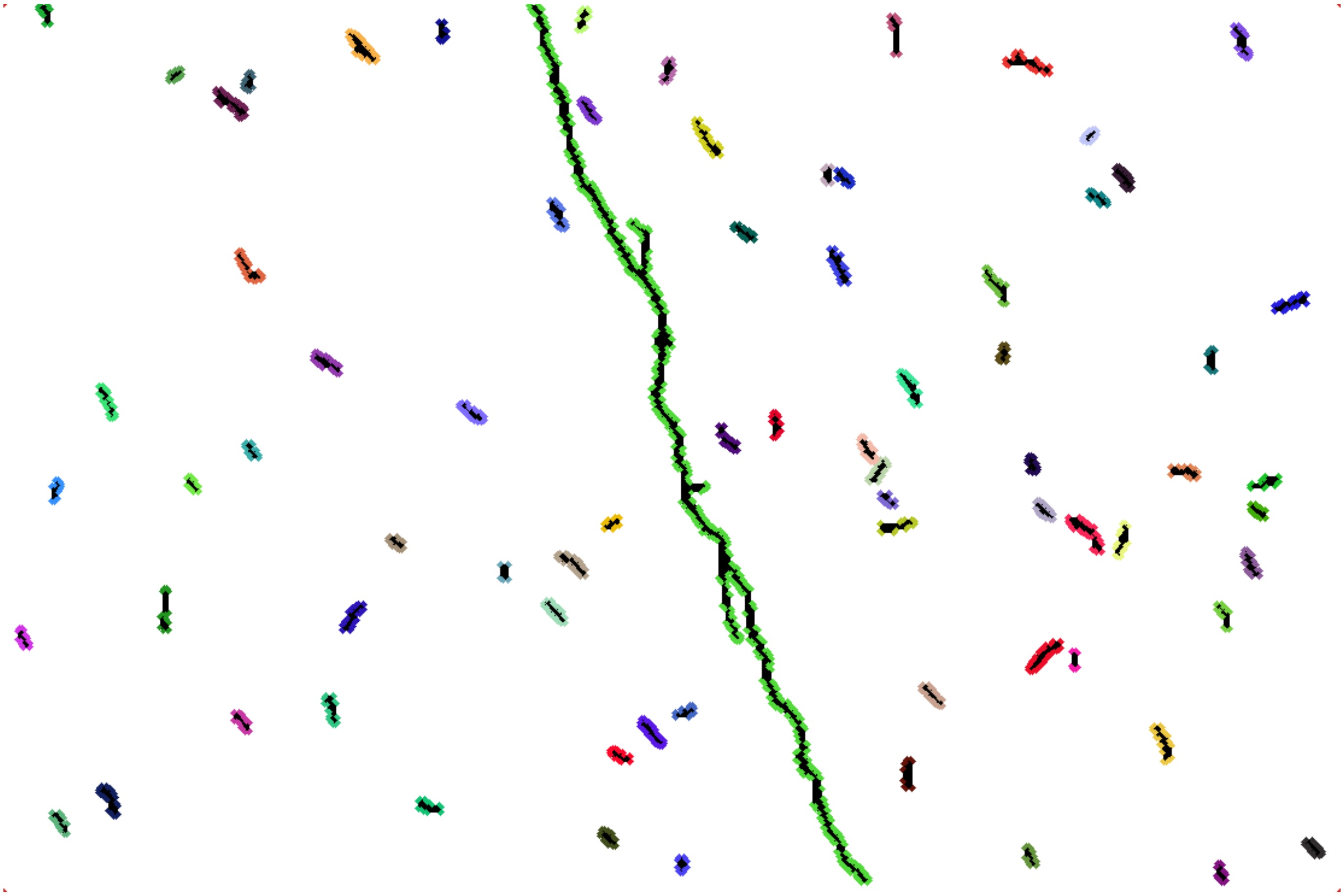
Thresholds used:

Filtered out the contours whose length is greater than 130, the actual length of the crack is around 5000 pixels

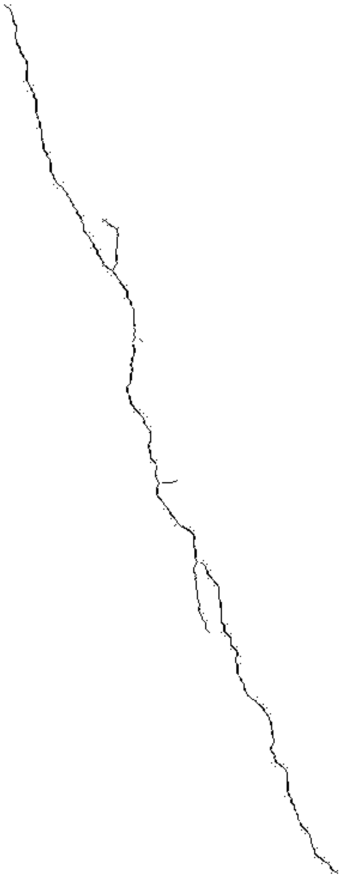
Applied two erosions and 1 dilation.



wall1-blobs.png



Walls1-contours.png



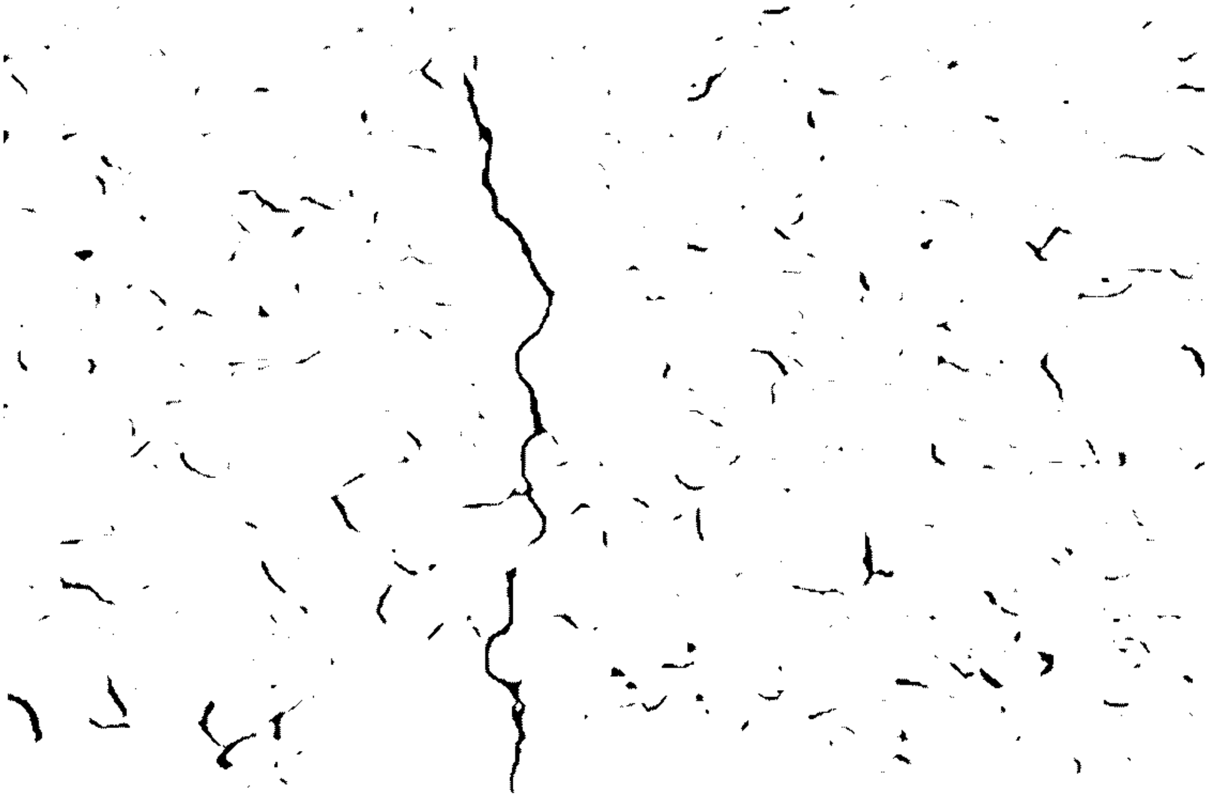
walls1-crack.png

Wall-2.png

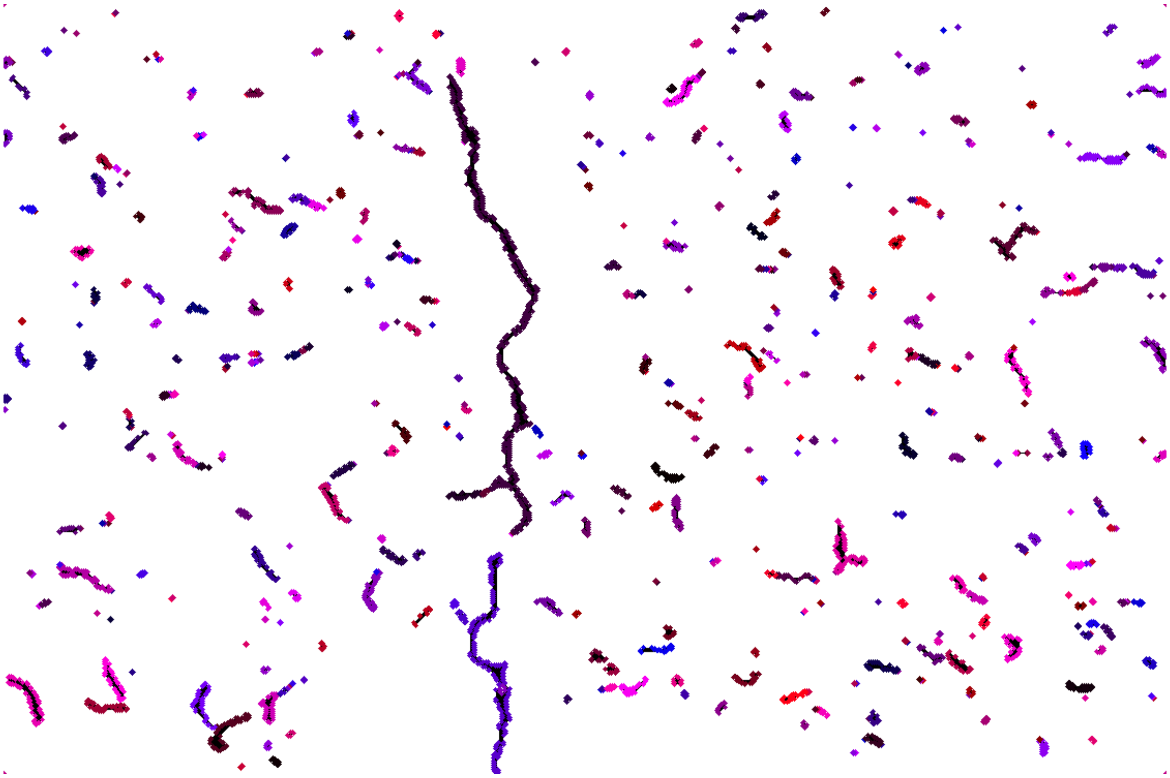
Thresholds used:

Filtered out the contours whose length is greater than 500, the actual total length of the crack is around 5000 pixels

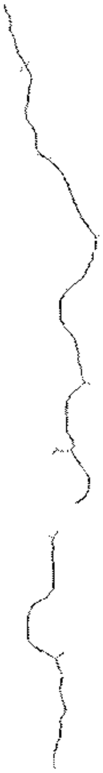
Applied 1 erosion and 2 dilations, using MORPH_ELIPSE



Wall1-blobs.png



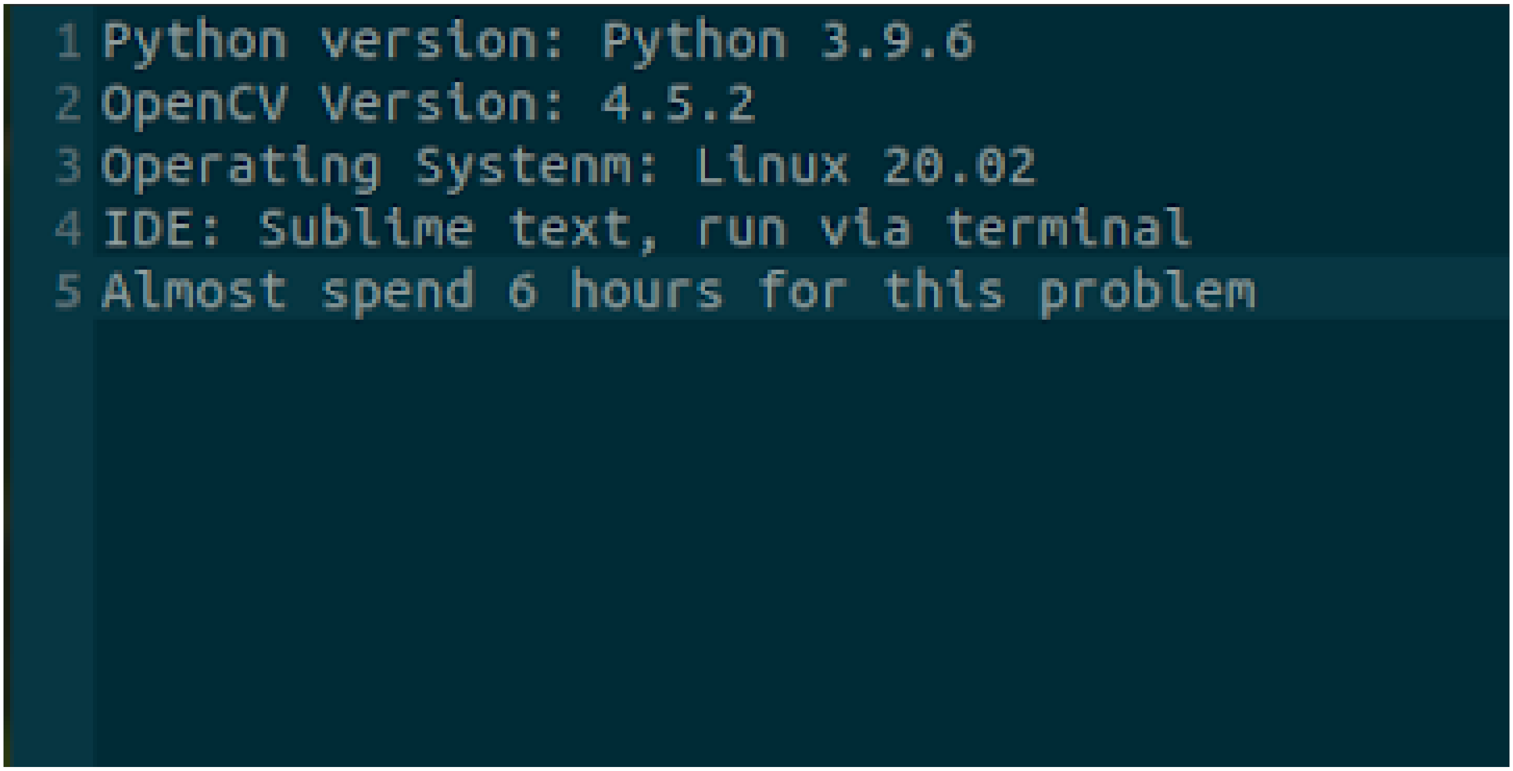
walls2-contours.png



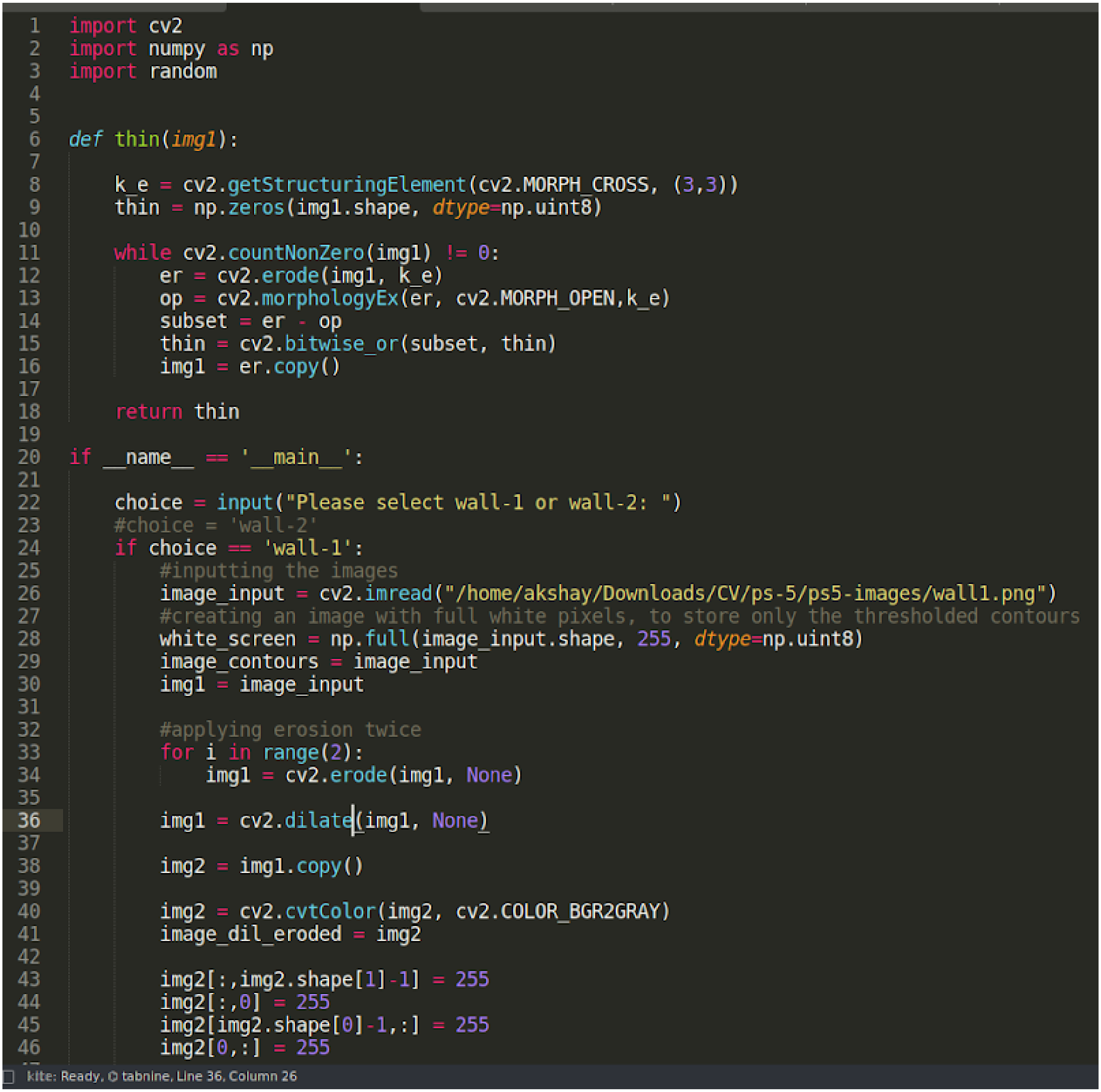
Walls2-crack.png

<u>Screenshots of code files</u>

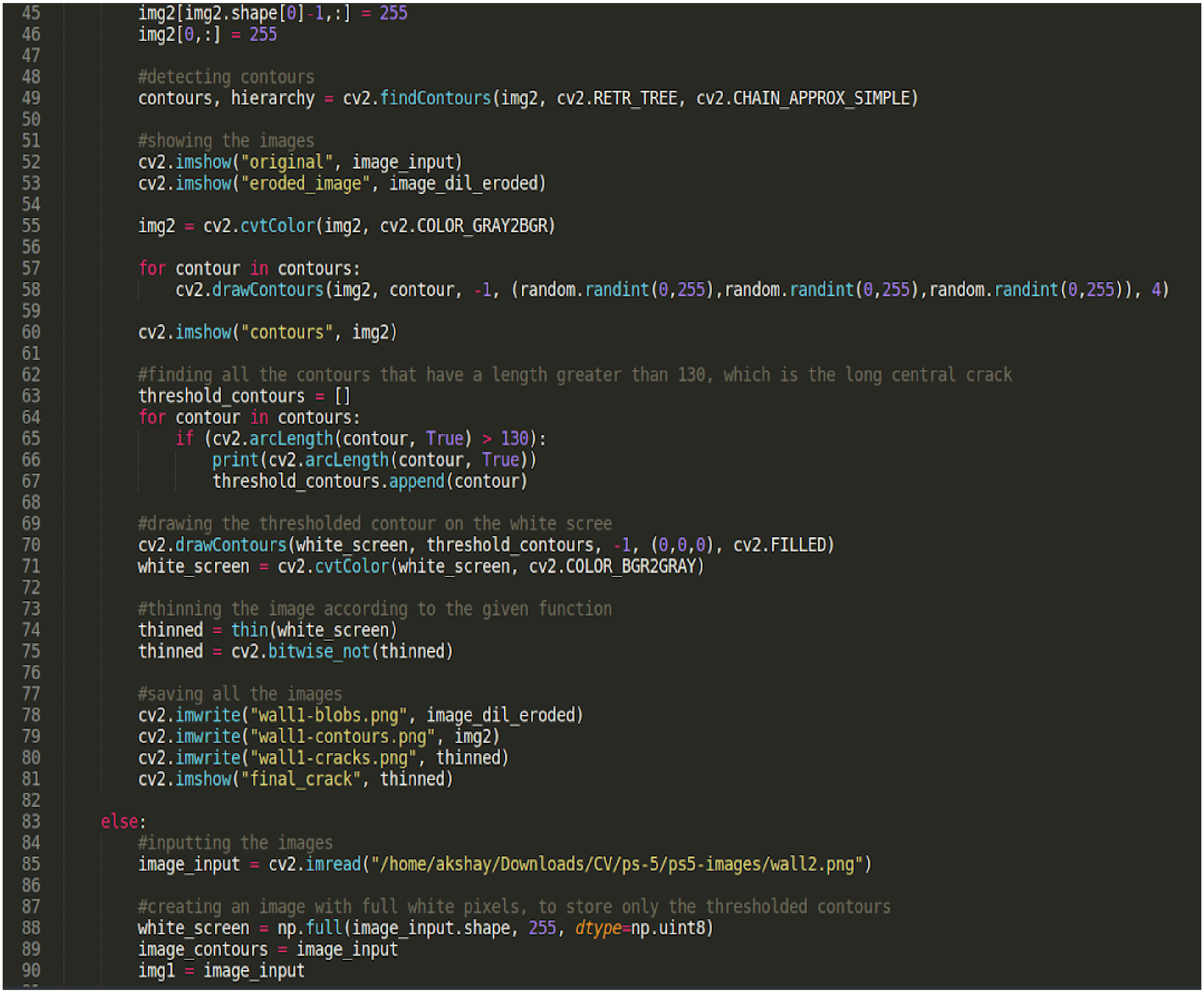
Readme.txt



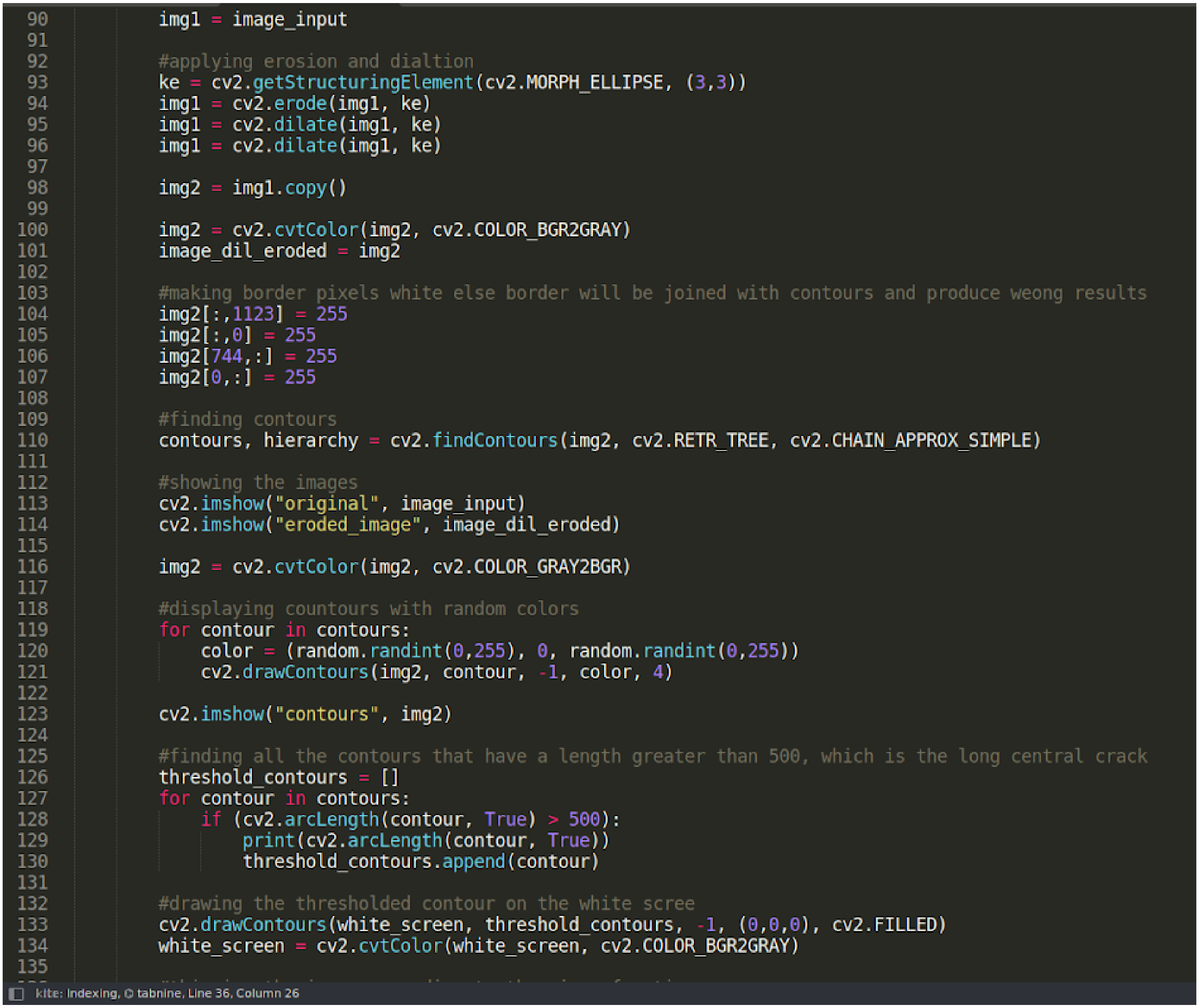
```
Python version: Python 3.9.6
OpenCV Version: 4.5.2
Operating Systenm: Linux 20.02
IDE: Sublime text, run via terminal
Almost spend 6 hours for this problem
```



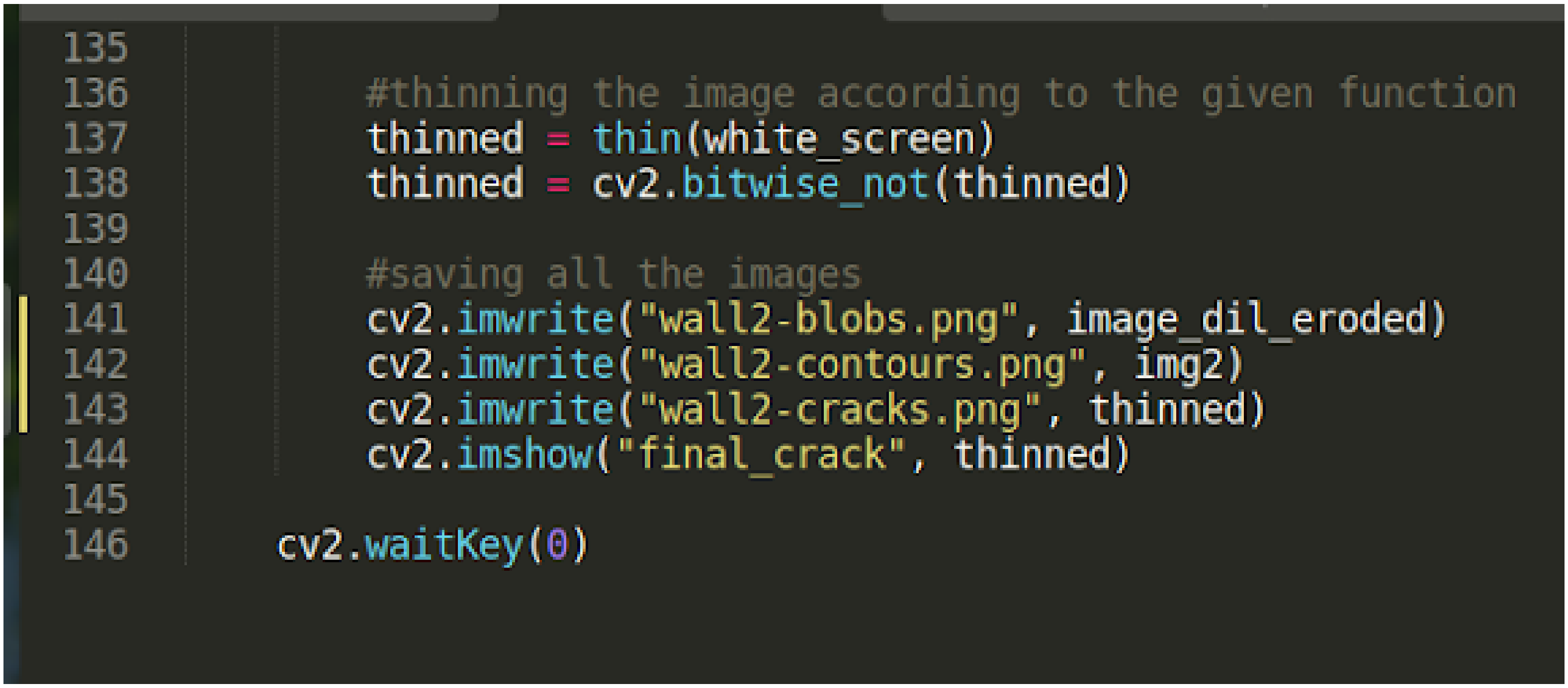
```python
import cv2
import numpy as np
import random


def thin(img1):

    k_e = cv2.getStructuringElement(cv2.MORPH_CROSS, (3,3))
    thin = np.zeros(img1.shape, dtype=np.uint8)

    while cv2.countNonZero(img1) != 0:
        er = cv2.erode(img1, k_e)
        op = cv2.morphologyEx(er, cv2.MORPH_OPEN,k_e)
        subset = er - op
        thin = cv2.bitwise_or(subset, thin)
        img1 = er.copy()

    return thin

if __name__ == '__main__':

    choice = input("Please select wall-1 or wall-2: ")
    #choice = 'wall-2'
    if choice == 'wall-1':
        #inputting the images
        image_input = cv2.imread("/home/akshay/Downloads/CV/ps-5/ps5-images/wall1.png")
        #creating an image with full white pixels, to store only the thresholded contours
        white_screen = np.full(image_input.shape, 255, dtype=np.uint8)
        image_contours = image_input
        img1 = image_input

        #applying erosion twice
        for i in range(2):
            img1 = cv2.erode(img1, None)

        img1 = cv2.dilate(img1, None)

        img2 = img1.copy()

        img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
        image_dil_eroded = img2

        img2[:,img2.shape[1]-1] = 255
        img2[:,0] = 255
        img2[img2.shape[0]-1,:] = 255
        img2[0,:] = 255
```



```
        img2[img2.shape[0]-1,:] = 255
        img2[0,:] = 255

        #detecting contours
        contours, hierarchy = cv2.findContours(img2, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

        #showing the images
        cv2.imshow("original", image_input)
        cv2.imshow("eroded_image", image_dil_eroded)

        img2 = cv2.cvtColor(img2, cv2.COLOR_GRAY2BGR)

        for contour in contours:
            cv2.drawContours(img2, contour, -1, (random.randint(0,255),random.randint(0,255),random.randint(0,255)), 4)

        cv2.imshow("contours", img2)

        #finding all the contours that have a length greater than 130, which is the long central crack
        threshold_contours = []
        for contour in contours:
            if (cv2.arcLength(contour, True) > 130):
                print(cv2.arcLength(contour, True))
                threshold_contours.append(contour)

        #drawing the thresholded contour on the white scree
        cv2.drawContours(white_screen, threshold_contours, -1, (0,0,0), cv2.FILLED)
        white_screen = cv2.cvtColor(white_screen, cv2.COLOR_BGR2GRAY)

        #thinning the image according to the given function
        thinned = thin(white_screen)
        thinned = cv2.bitwise_not(thinned)

        #saving all the images
        cv2.imwrite("wall1-blobs.png", image_dil_eroded)
        cv2.imwrite("wall1-contours.png", img2)
        cv2.imwrite("wall1-cracks.png", thinned)
        cv2.imshow("final_crack", thinned)

    else:
        #inputting the images
        image_input = cv2.imread("/home/akshay/Downloads/CV/ps-5/ps5-images/wall2.png")

        #creating an image with full white pixels, to store only the thresholded contours
        white_screen = np.full(image_input.shape, 255, dtype=np.uint8)
        image_contours = image_input
        img1 = image_input
```



```
        img1 = image_input

        #applying erosion and dialtion
        ke = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3,3))
        img1 = cv2.erode(img1, ke)
        img1 = cv2.dilate(img1, ke)
        img1 = cv2.dilate(img1, ke)

        img2 = img1.copy()

        img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
        image_dil_eroded = img2

        #making border pixels white else border will be joined with contours and produce weong results
        img2[:,1123] = 255
        img2[:,0] = 255
        img2[744,:] = 255
        img2[0,:] = 255

        #finding contours
        contours, hierarchy = cv2.findContours(img2, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

        #showing the images
        cv2.imshow("original", image_input)
        cv2.imshow("eroded_image", image_dil_eroded)

        img2 = cv2.cvtColor(img2, cv2.COLOR_GRAY2BGR)

        #displaying countours with random colors
        for contour in contours:
            color = (random.randint(0,255), 0, random.randint(0,255))
            cv2.drawContours(img2, contour, -1, color, 4)

        cv2.imshow("contours", img2)

        #finding all the contours that have a length greater than 500, which is the long central crack
        threshold_contours = []
        for contour in contours:
            if (cv2.arcLength(contour, True) > 500):
                print(cv2.arcLength(contour, True))
                threshold_contours.append(contour)

        #drawing the thresholded contour on the white scree
        cv2.drawContours(white_screen, threshold_contours, -1, (0,0,0), cv2.FILLED)
        white_screen = cv2.cvtColor(white_screen, cv2.COLOR_BGR2GRAY)
```



```

        #thinning the image according to the given function
        thinned = thin(white_screen)
        thinned = cv2.bitwise_not(thinned)

        #saving all the images
        cv2.imwrite("wall2-blobs.png", image_dil_eroded)
        cv2.imwrite("wall2-contours.png", img2)
        cv2.imwrite("wall2-cracks.png", thinned)
        cv2.imshow("final_crack", thinned)

    cv2.waitKey(0)
```