CVE Assignment 5 Report

Question 1 (PS5-1)

Algorithm and Observations

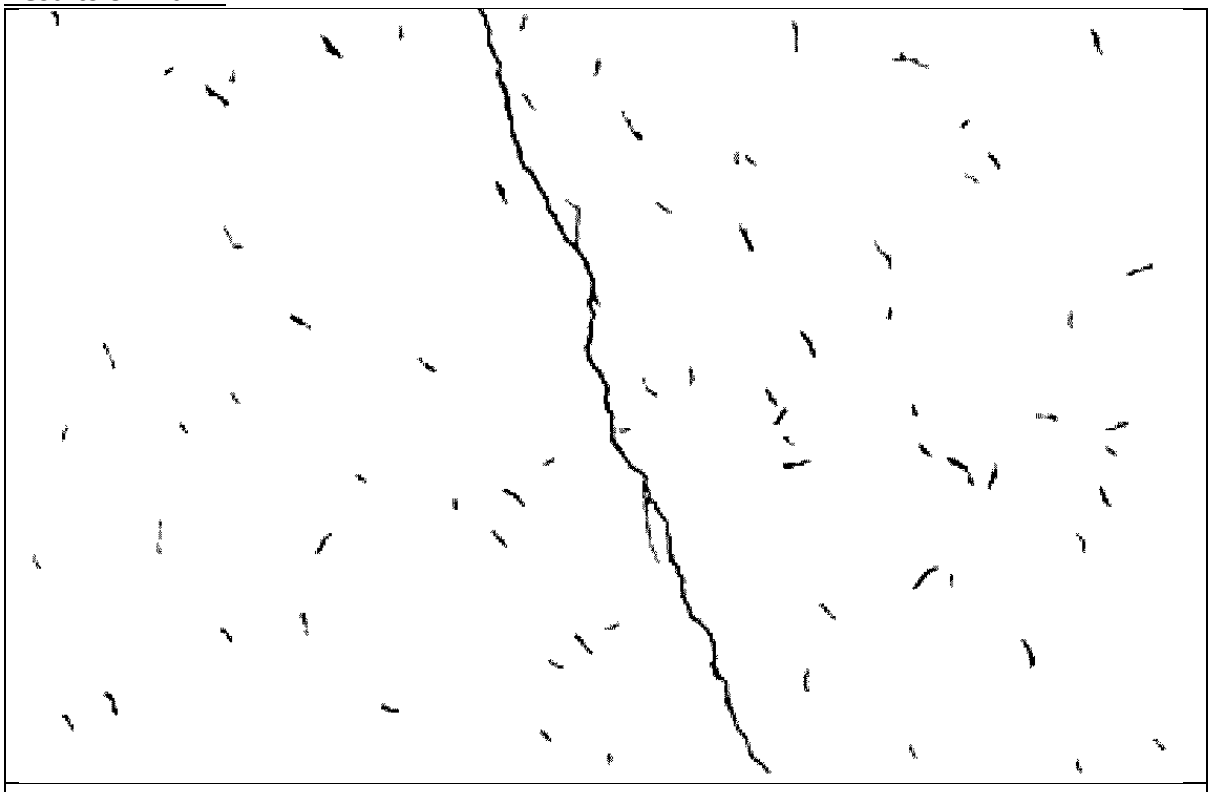
The objective of this assignment was to identify blobs in a binary image and highlight the crack features present in those blobs. Binarising an image is an optimal way of going forward to solve this problem. Thresholded images allow us to easily classify between interested regions and background in an image. When identifying charecteristics features present in an image, thresholding highlights the region of interest.

Four functions were used to help solve this problem, each called in a predefined order. The details of these functions are given below.

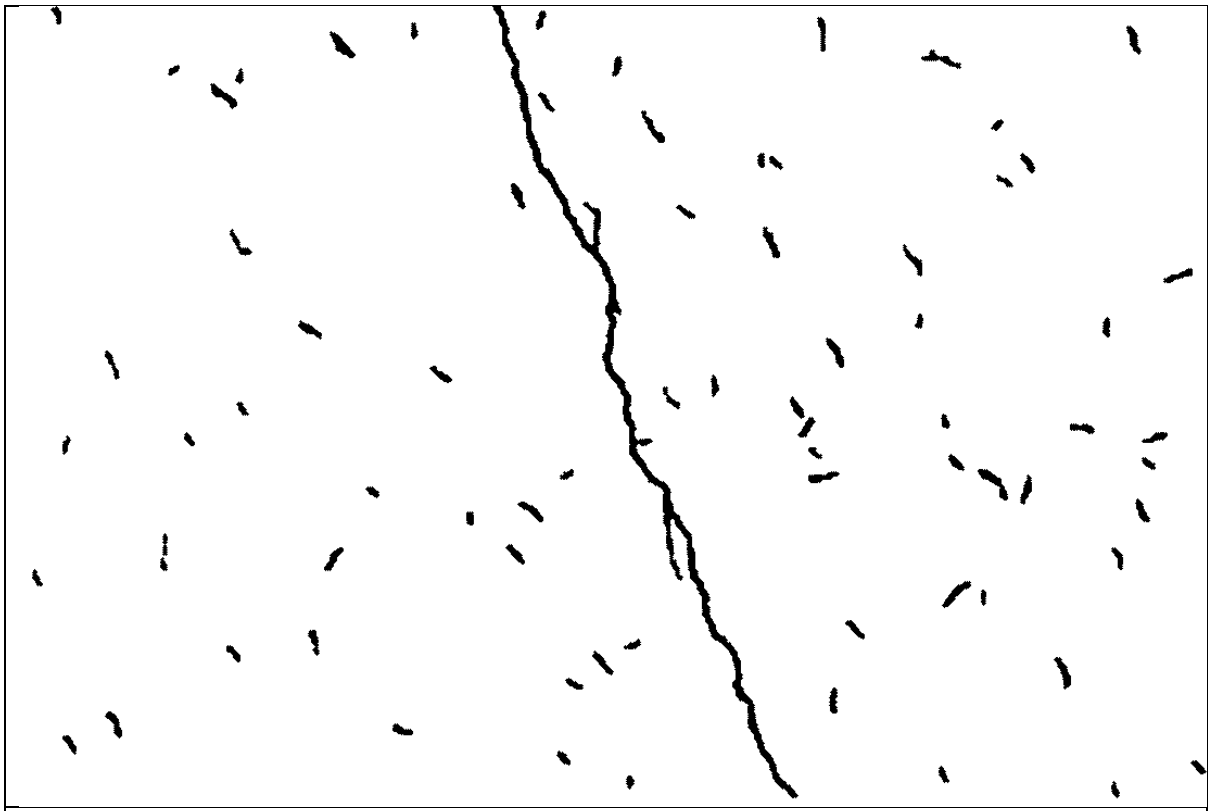
1. erode_dilate() : The objective of this function, as the name suggests, is to perform erode and dilate functions on the image with the aim to produce clear and identifiable blobs within the image. The method of applying two erode functions successively followed by a single dilate produced a good result for our requirements. A 3x3 cross shaped kernel was used for our purposes.
2. cont_detect(): The objective of this function was to find and draw the contours of the produced blobs. The grayscale image was thresholded and passed through the findContours() function. The outout contours were drawn in random colors on the original image.
3. thresholding(): The objective of this function was to threshold the image and segregate the non crack features from the actual cracks. The contour area parameter of each contour was used to classify the blob as a crack or not. From the training images provided, a contour area of 2000 was found optimal. A problem encountered was handling the blobs on the image edges with open contours. To solve this, a white border of thickness 3 was introduced before contour detection and blob thresholding. The final thresholded image was then cropped to the original size.
4. central_axis(): This function aimed at thinning the final image. A set of erode and dilate function reduced the original image in thickness until no difference was visible in the eroded and dilated images. This allowed us to find the central axis or the “spine” of the crack.

Each of these functions were called in a predefined manner to produce the results shown below.

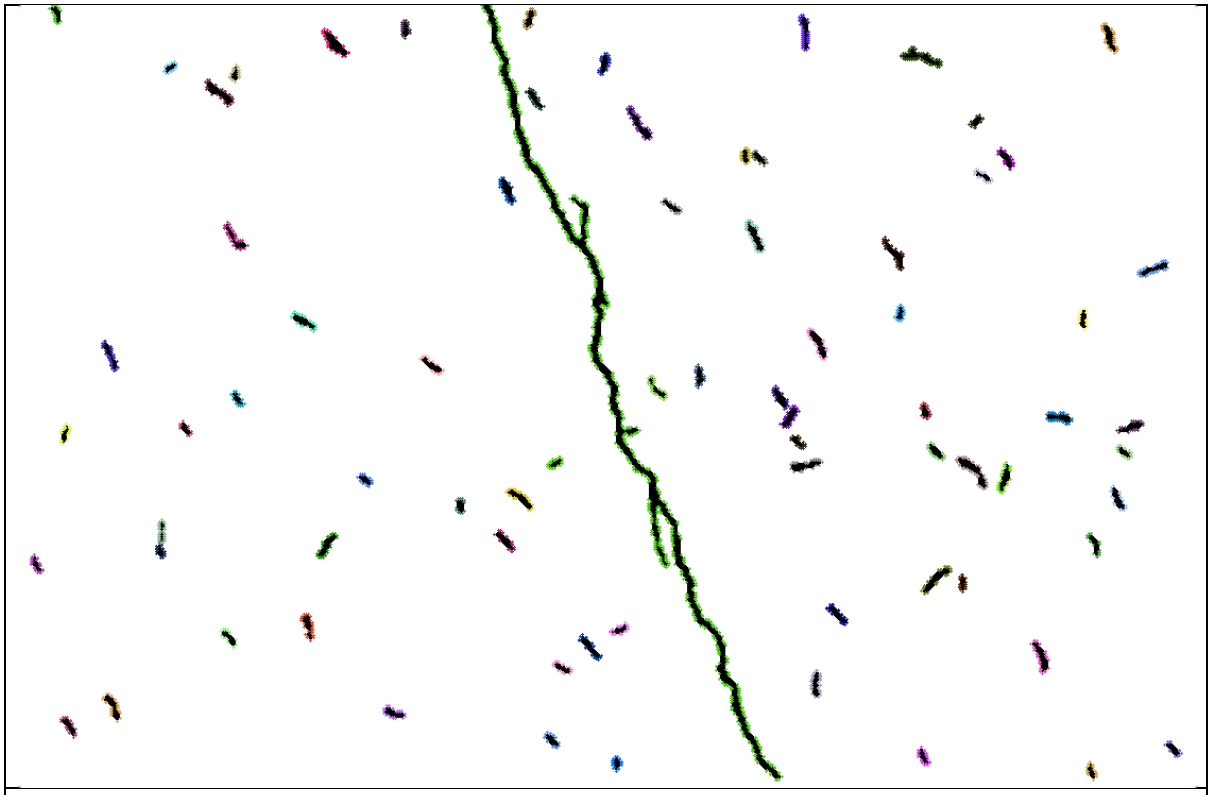
Results of Wall 1



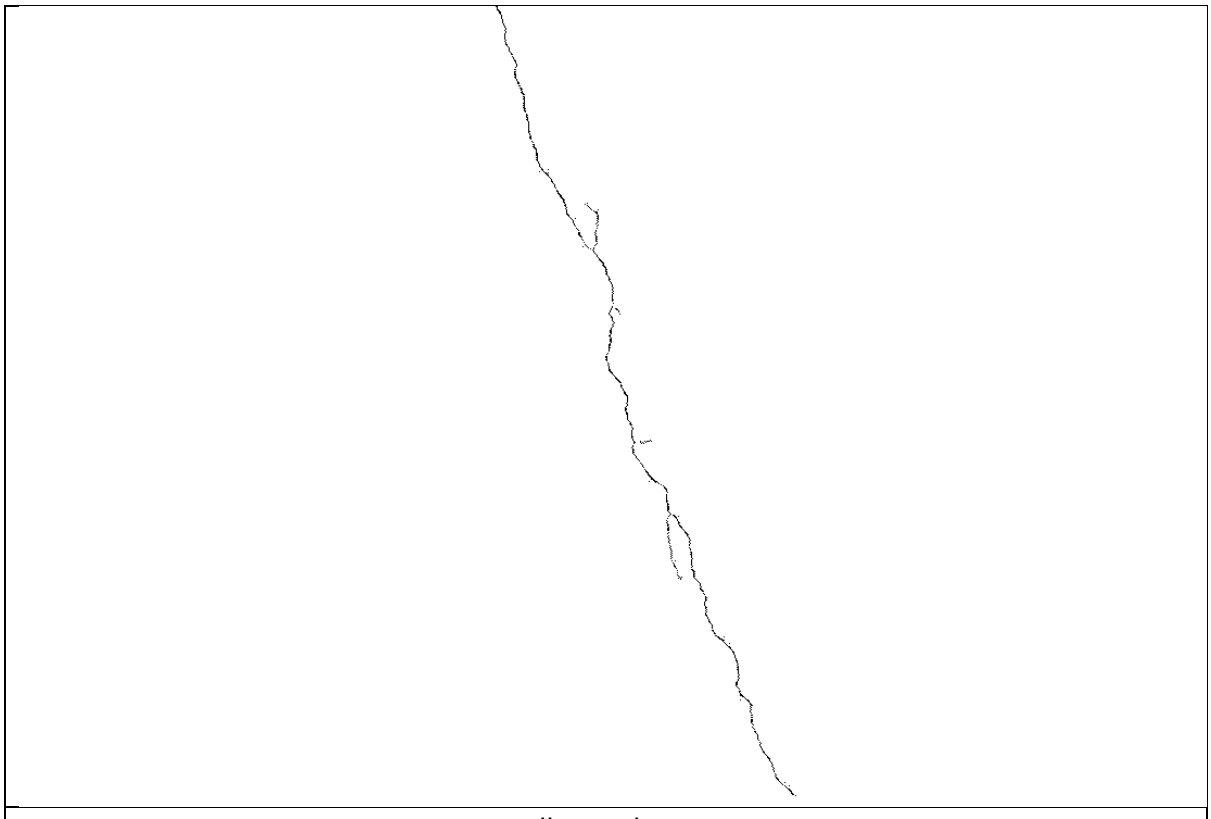
Wall1.png (Original)



Wall1-blob.png

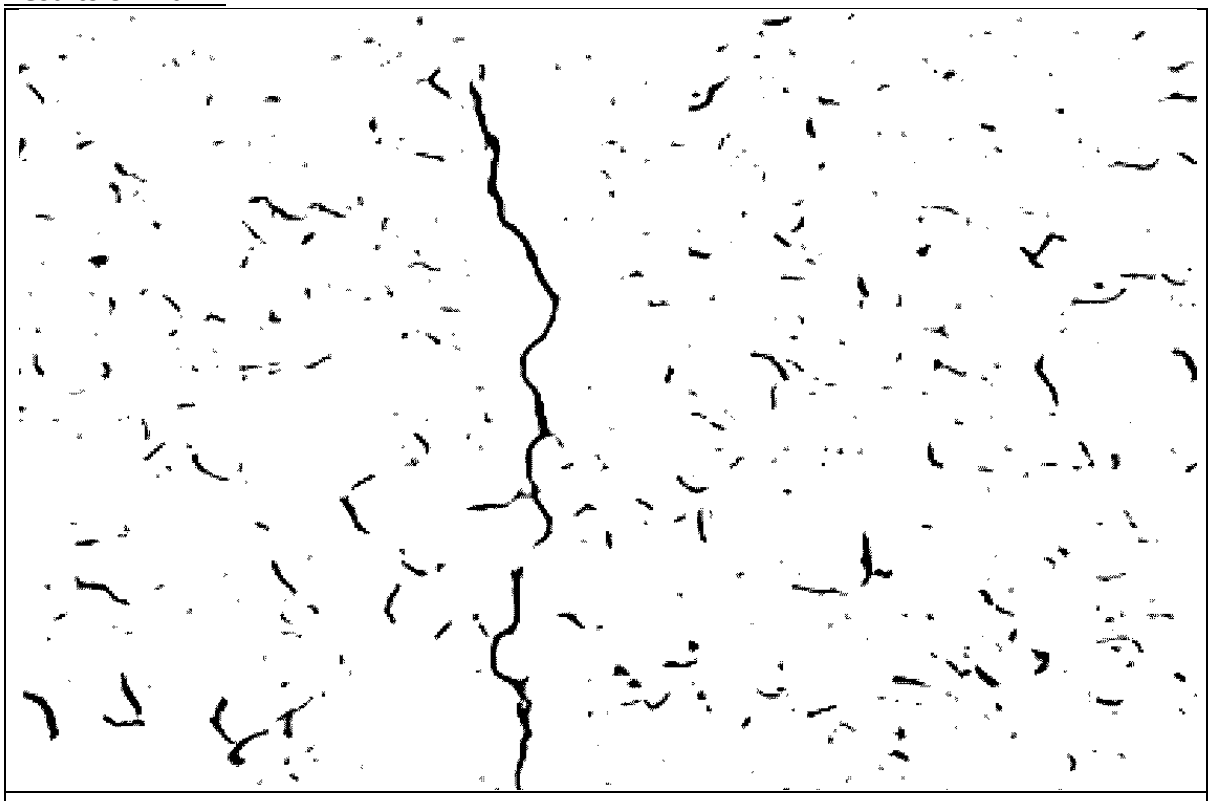


Wall1-contours.png

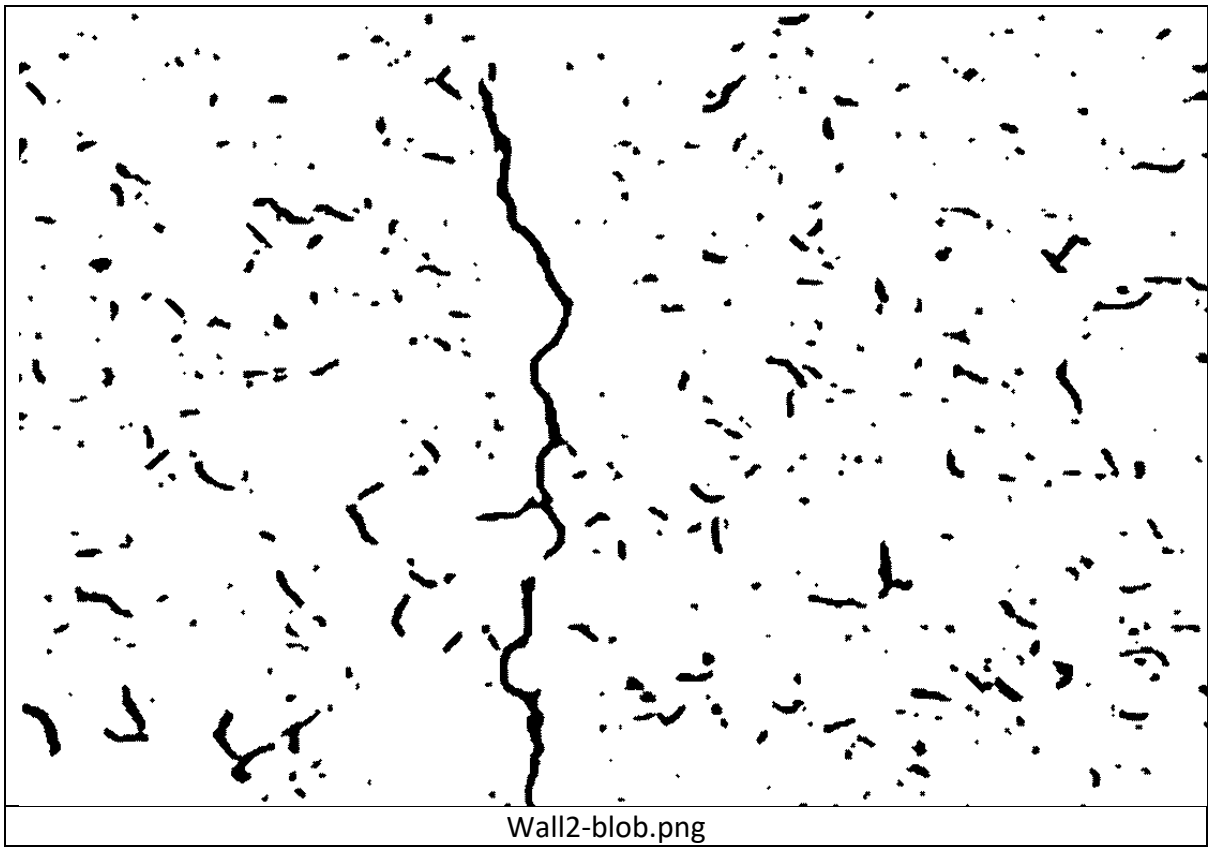


Wall1-cracks.png

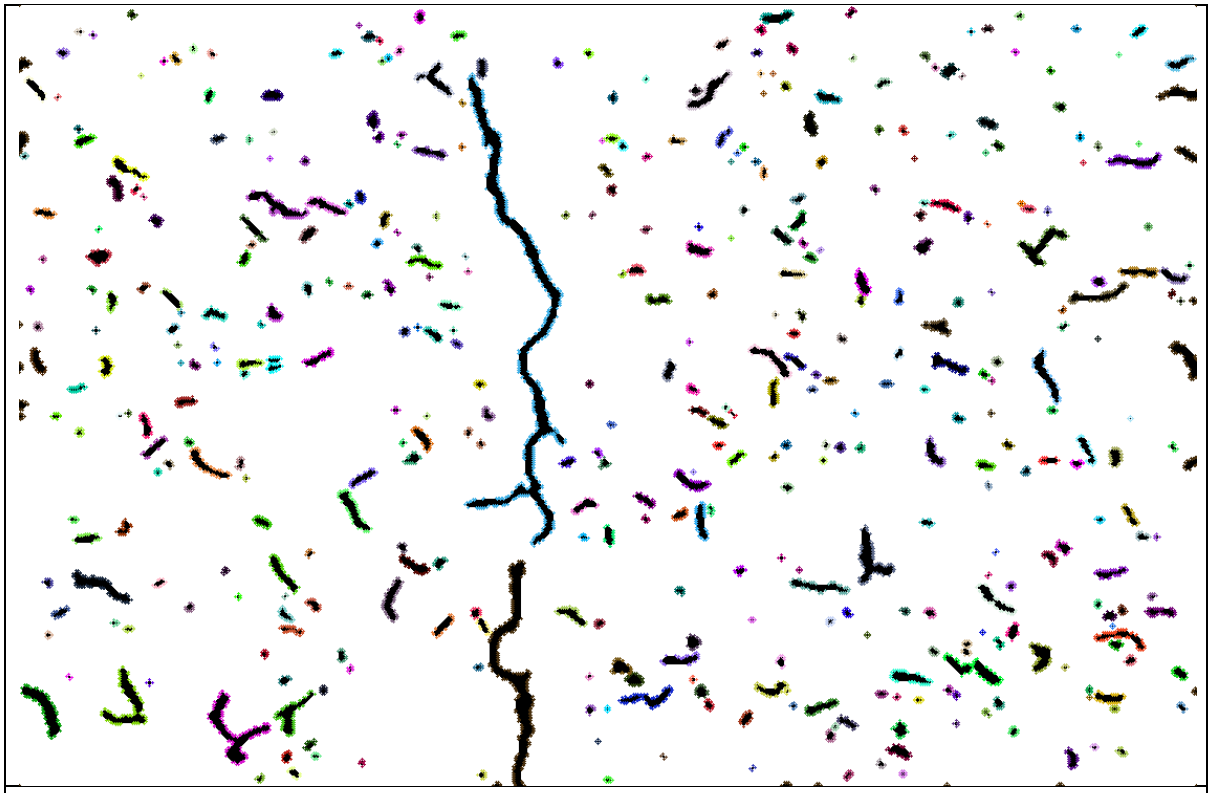
Results of Wall 2



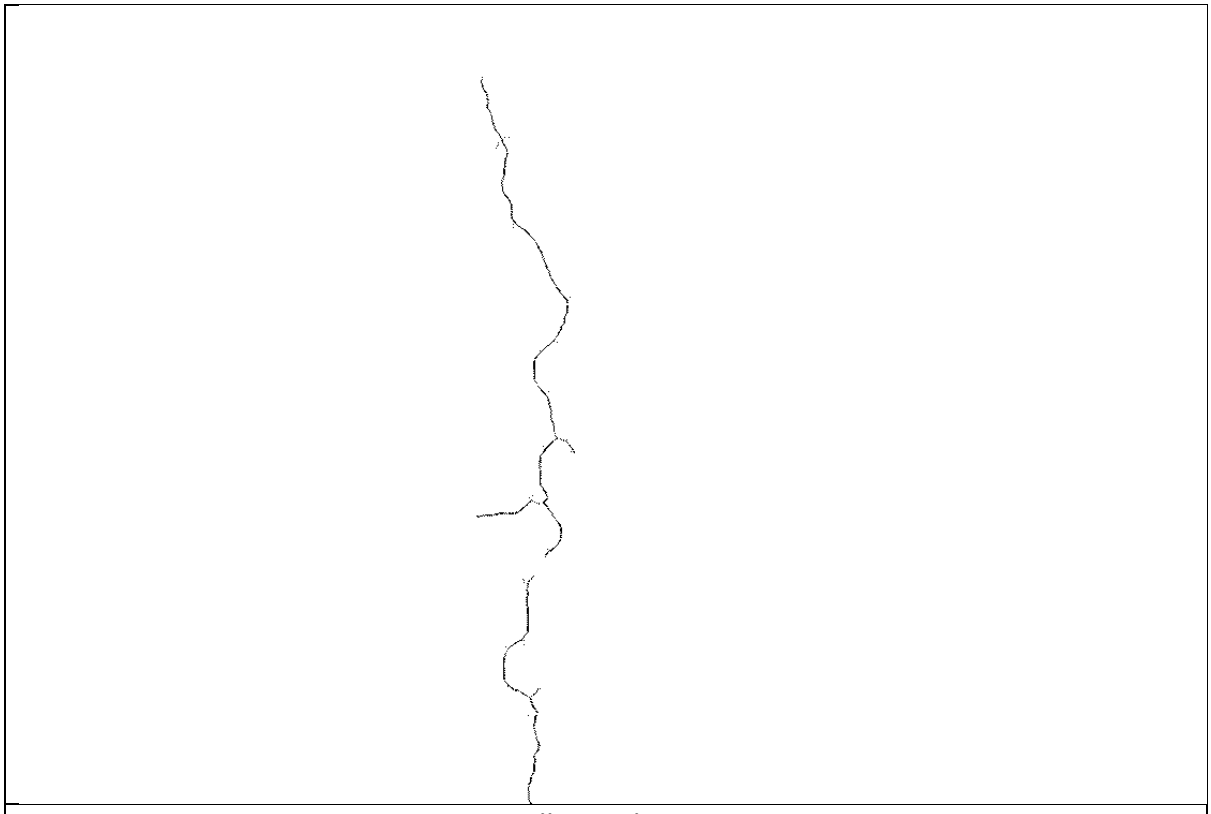
Wall2.png (Original)



Wall2-blob.png



Wall2-contours.png



Wall2-cracks.png

Source Code

```
# Sunday, 29th October, 2022
# Problem Set 5 - Computer Vision for Engineers

import cv2 as cv
import numpy as np

def erode_dilate(img, fname): #Erode and Dilate Functions
    k_e = cv.getStructuringElement(cv.MORPH_CROSS, (3,3))
    img = cv.erode(img, k_e)
    img = cv.erode(img, k_e)
    img = cv.dilate(img, k_e)

    cv.imwrite(fname[0]+"-blob."+fname[1], img)
    return img

def cont_detect(img, fname):    #Contour detection and drawing
    icopy = img.copy()
    img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    ret, thresh = cv.threshold(img_gray, 127, 255, 0)
    contours, hierarchy = cv.findContours(thresh, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
    for i in range(len(contours)):
        color = list(np.random.random(size = 3) * 255) #For random colors
        icopy = cv.drawContours(icopy, contours[i], -1, color,  2)

    cv.imwrite(fname[0]+"-contours."+fname[1], icopy)
    return icopy

def thresholding(img, fname):
    img = cv.copyMakeBorder(img, 3,3,3,3, cv.BORDER_CONSTANT, value=[255,255,255]) # For edge
contours

    img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    ret, thresh = cv.threshold(img_gray, 127, 255, 0)
    contours, hierarchy = cv.findContours(thresh, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
    threshold_blob = 2000

    for i in range(len(contours)):
        cnt = contours[i]
        area = cv.contourArea(cnt)
        if area <= threshold_blob:  #Thresholding
            thresh = cv.drawContours(thresh, [cnt], -1, 255, -1)    #Filling contour with white color
    thresh = thresh[3:-1-2, 3:-1-2] #Cropping to initial size
    cv.imwrite(fname[0]+"-thresholded."+fname[1], thresh)
```

```
    return thresh

def central_axis(img, fname):
    img = cv.bitwise_not(img)
    fimage = img.copy()
    k_e = cv.getStructuringElement(cv.MORPH_CROSS, (3,3))
    thin = np.zeros(img.shape, dtype='uint8')
    while(np.sum(fimage)!=0):
        cimg = cv.erode(fimage, k_e)
        oimg = cv.morphologyEx(cimg, cv.MORPH_OPEN, k_e)
        subset = cimg - oimg
        thin = cv.bitwise_or(subset, thin)  #union of previous data and new thiined features
        fimage = cimg.copy()
    thin = cv.bitwise_not(thin)
    cv.imwrite(fname[0]+"-cracks."+fname[1], thin)
    return thin

fname = input("Enter Image Path: ")
img = cv.imread(fname)
fname = fname.split('.')

erode_dilate_img = erode_dilate(img, fname)
cv.imshow("Eroded and Dilated", erode_dilate_img)
cv.waitKey(100)

cont_detect_img = cont_detect(erode_dilate_img, fname)
cv.imshow("Contour Detected Image", cont_detect_img)
cv.waitKey(100)

thresholded_img = thresholding(erode_dilate_img, fname)
cv.imshow("Thresholded Image", thresholded_img)
cv.waitKey(100)

crack_image = central_axis(thresholded_img,fname)
cv.imshow("Central Axis Image", crack_image)
cv.waitKey(0)

cv.destroyAllWindows()
```

System Specifications

Operating System: macOS Monterey Version 12.5.1
Hardware: MacBook Air 2017 (Intel Core i5)
Python: Conda environment utilizing Python 3.9.1
IDE: Visual Studio Code