CVE Assignment 2 Report

Algorithm

The idea behind the problem involves identifying the brightest region in the image and identify that area using an indicator in the pseudo color representation of the image. A lookup table mapping the intensity value of each pixel to the corresponding blue, green and red value is required for the creation of the pseudo color. A conditional search for the index values of pixels with maximum intensities is used to determine the maximum intensity area, and subsequently determine the center of gravity of the area.

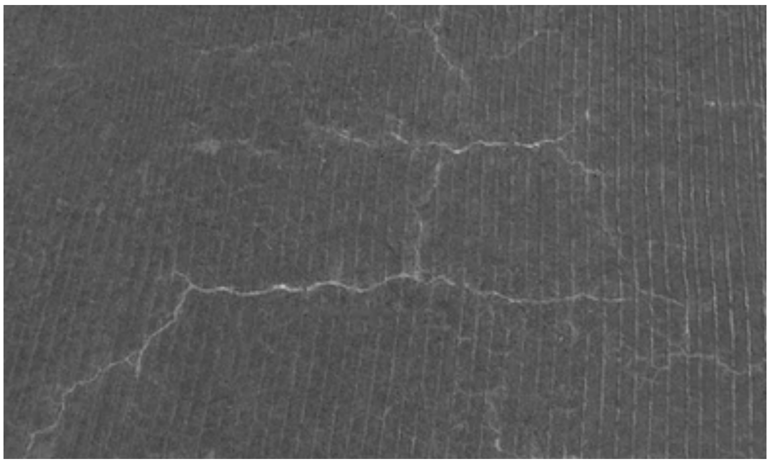
The following steps describe the algorithm for solving the problem

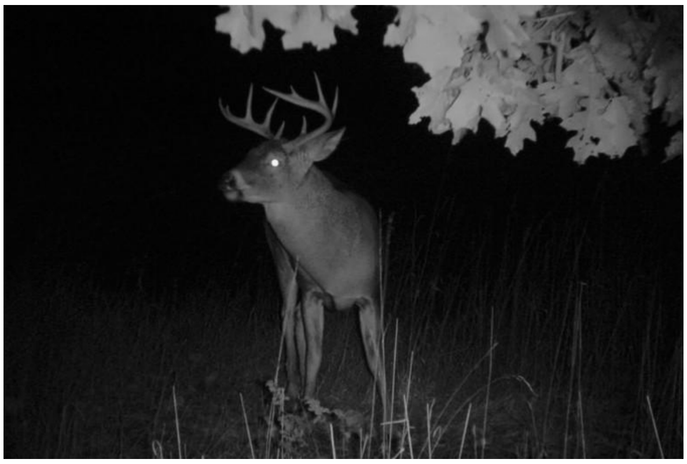
1. Input an image from the user and convert it to the grayscale representation using cv2. cvtColor() function.
2. Generate a lookup table for the input image. Details of the lookup table generation are given below.
3. Use list comprehension to generate the blue, green and red channels for the image by mapping the pixel intensity to the channel intensity in the lookup table. Merge channels to create the output pseudo color image using cv2.merge() function.
4. Identify the pixel indexes with the maximum intensity value and calculate the center of gravity for the indices. The function where(), amax() and amin() from the numpy library are used for this. The returned data is used to find the central index for the area. Create a circle and cross hairs at the centrepoint of that area using the cv2.circle() and cv2.line function.
5. Display the output image

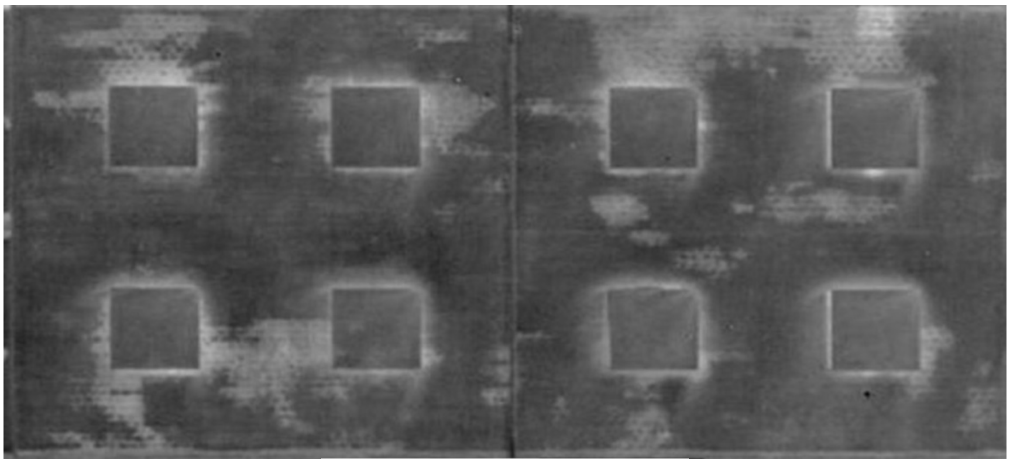
Creating the lookup table:

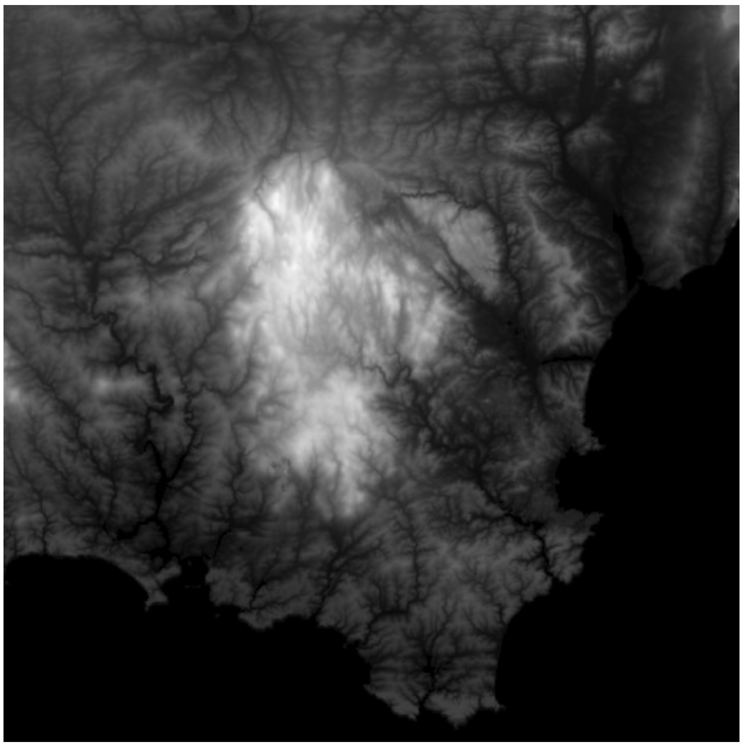
1. Identify the max and min values if the intensity using the numpy amax() and amin() function.
2. Divide the range between the max and min values into four equal sectors.
    a. Blue channel is highest in the first sector; descends linearly to 0 in the second sector; and remains zero in the remaining sectors.
    b. Green Channel ascends from 0 to highest value in the first sector; remains there for the next two sectors; and slopes down to 0 in the last sector.
    c. The red channel is 0 in the first two sectors; ascends to the highest value in the third sector; and remains stable in the last sector.
3. The function np.empty() was used to create an empty lookup table of the dimensions (3,256), with one from for each channel.
4. Data from the step 2 was used to fill the lookup table, using the np.linspace() function to define the linear ascent and descent.

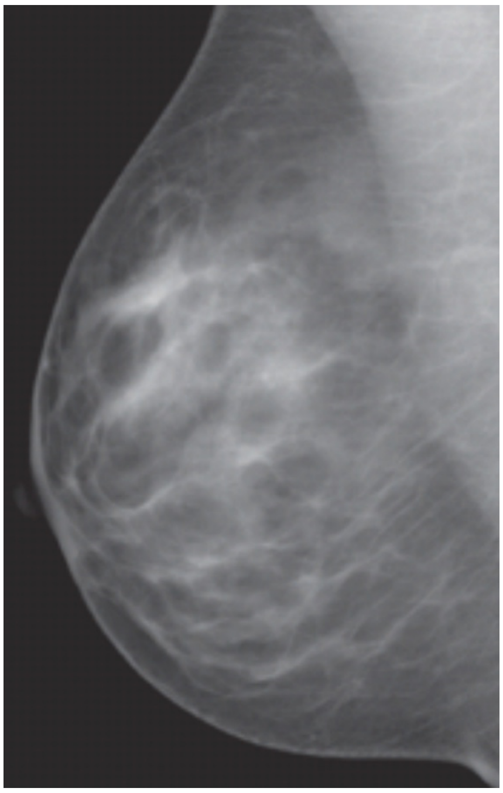
Input Images


cracks.png

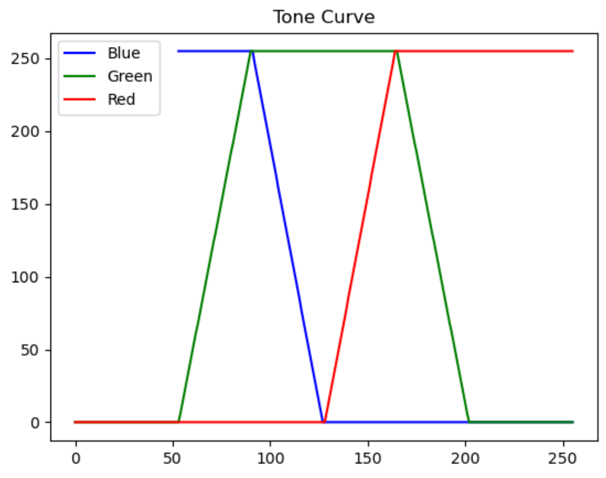

night-vision.png


thermal.png

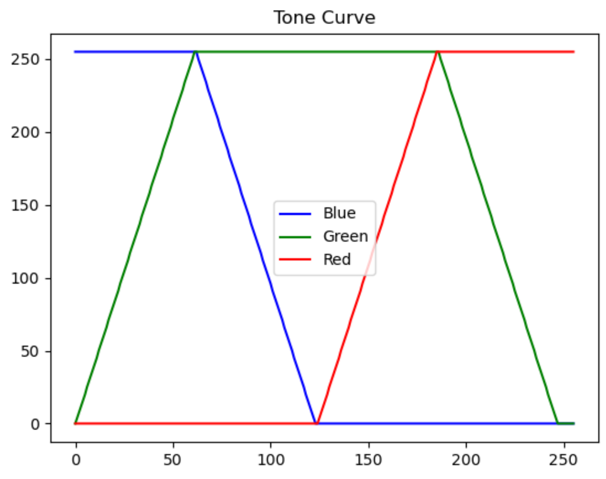

topography.png


x-ray.png

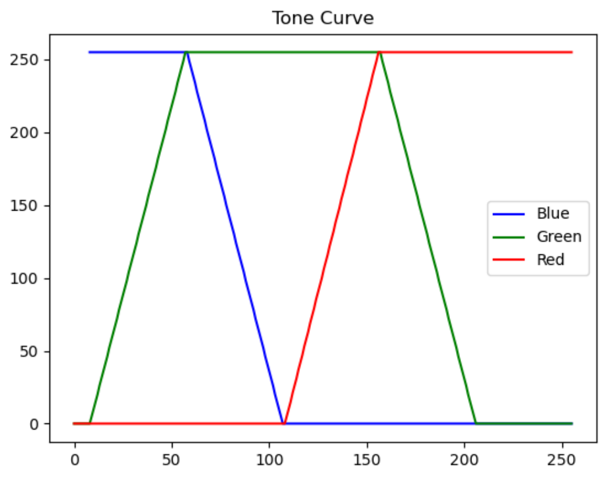
Tone Curve for Images



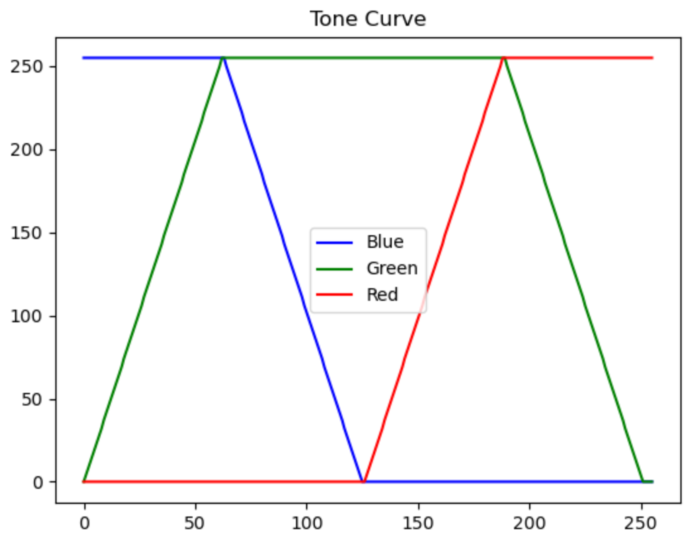
cracks-tone curve.png



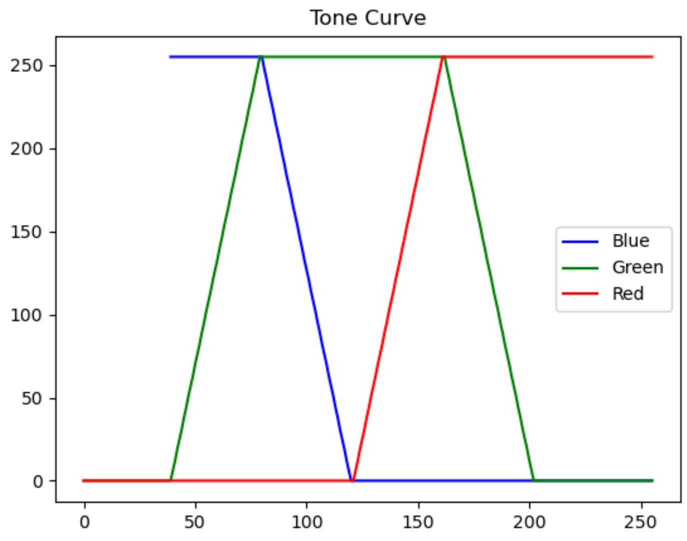
night-vision-tone curve.png



thermal-tone curve.png

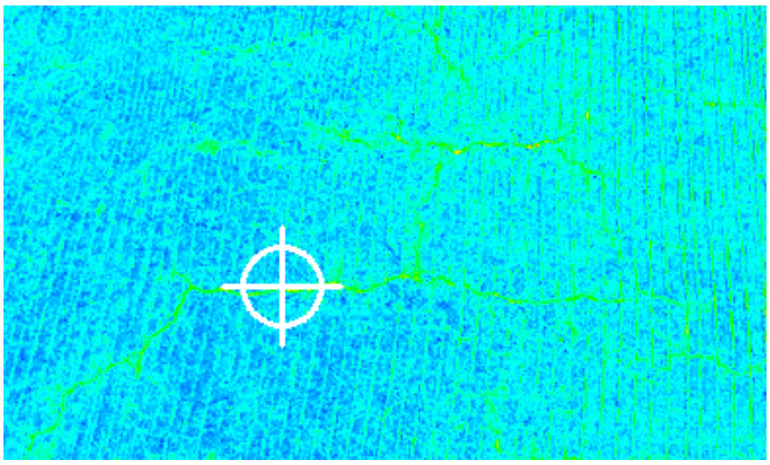


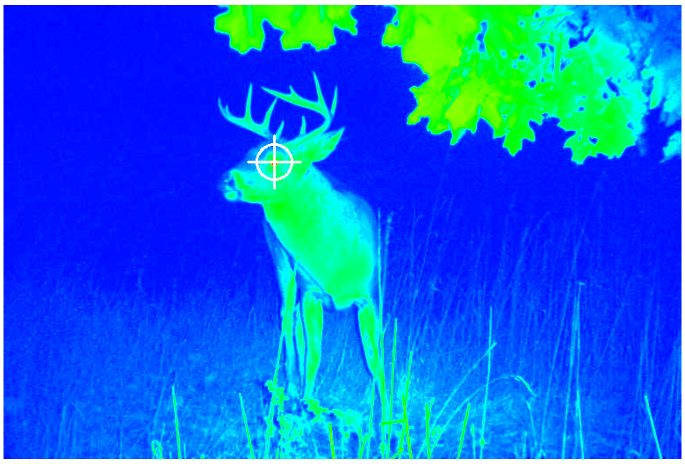
topography-tone curve.png

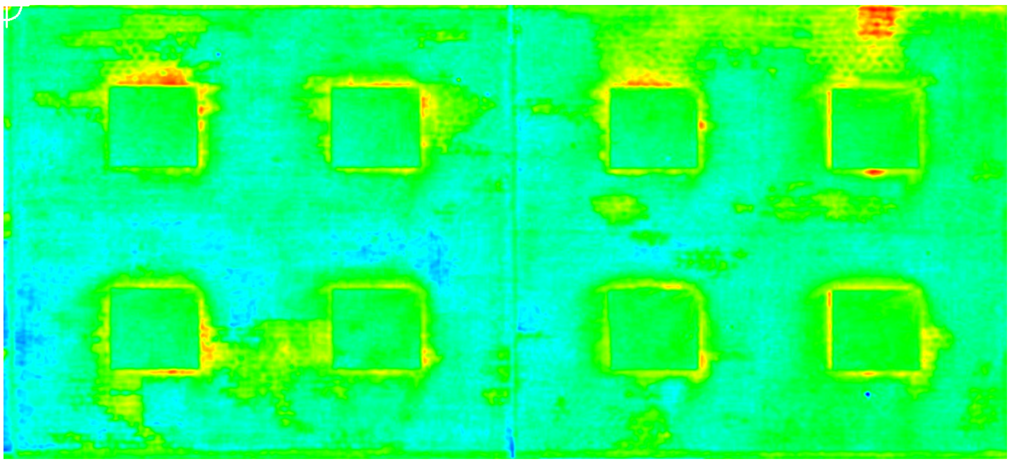


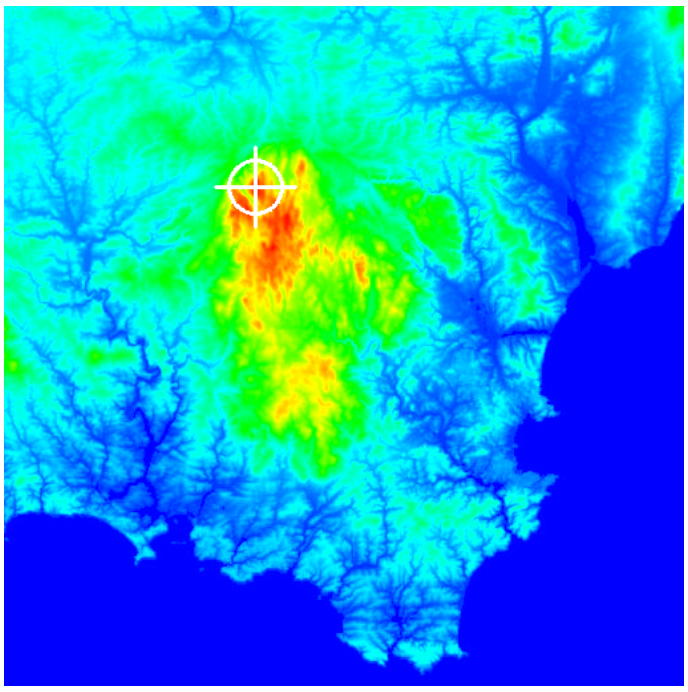
x-ray-tone curve.png

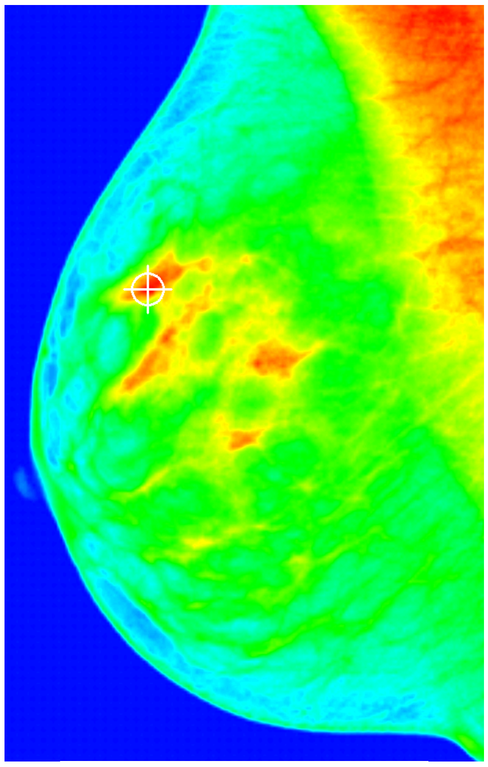
Output Images


cracks-color.png


night-vision-color.png


thermal-color.png


topography-color.png


x-ray-color.png

Source Code

```
#September 19th, 2022
#Aman Chulawala
#24678 PS2 Q1

from math import ceil
from matplotlib import pyplot as plt
import cv2 as cv
import numpy as np

#Convert input image to gray scale
def color2gray(img):
    gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    hist_gen(gray)
    return gray

#Generate Lookup array for the intensity range specified in the image
def lookup_gen(img):
    mini = np.amin(img)
    maxi = np.amax(img)
    q1 = mini + ceil((maxi-mini)/4)
    q2 = mini + ceil((maxi-mini)/2)
    q3 = maxi - ceil((maxi-mini)/4)
    lookup = np.zeros((3,256), dtype = 'uint8')
    #Generating Blue channel
    lookup[0,mini:q1] = 255
    lookup[0, q2:] = 0
    lookup[0, q1:q2] = np.linspace(255,0,num = q2-q1)
    #Generating Green Channel
    lookup[1,mini:q1] = np.linspace(0,255,num = q1-mini)
    lookup[1,q1:q3] = 255
    lookup[1, q3:maxi] = np.linspace(255,0, num = maxi-q3)
    #Generating Red Channel
    lookup[2, 0:q2] = 0
    lookup[2, q2:q3] = np.linspace(0,255, num = q3-q2)
    lookup[2, q3:] = 255
    #Plotting lookuptable
    plt.figure(2)
    plt.title('Tone Curve')
    plt.plot(np.linspace(mini,255, num = len(lookup[0,mini:])), 
lookup[0,mini:], label = "Blue", color = 'blue')
    plt.plot(np.linspace(0,255, num = 256), lookup[1,:], label = 
"Green", color = 'green')
```

```
    plt.plot(np.linspace(0,255, num = 256), lookup[2,:], label = "Red",
color = 'red')
    plt.legend()
    plt.show()
    return lookup, mini, maxi

#Plotting Histogram of intensities for reference
def hist_gen(img):
    histSize = 256
    histRange = (0, 256)
    hist_data = cv.calcHist(img, [0], None, [histSize], histRange,
accumulate=False)
    plt.figure(1)
    plt.title("Grayscale Histogram")
    plt.xlabel("Bins")
    plt.ylabel("# of Pixels")
    plt.plot(hist_data)
    plt.xlim([0, 256])

#Convert image to pseudocolored based on the lookup array
def convert(img, lookup):
    b_channel = [[lookup[0,img[i,j]] for j in range(0,len(img[0]))] for
i in range(0,len(img))]
    g_channel = [[lookup[1,img[i,j]] for j in range(0,len(img[0]))] for
i in range(0,len(img))]
    r_channel = [[lookup[2,img[i,j]] for j in range(0,len(img[0]))] for
i in range(0,len(img))]
    pseudocolored = cv.merge(np.array([b_channel, g_channel,
r_channel]))
    return pseudocolored

#Draw circle and cross hairs on the image
def draw(img, draw_img, mini, maxi):
    means = np.where(img == maxi)
    y_mean = round(np.mean(means[0]))
    x_mean = round(np.mean(means[1]))
    cv.circle(draw_img, (x_mean, y_mean), 20, (255,255,255), 2)
    cv.line(draw_img, (x_mean-30, y_mean), (x_mean+30, y_mean),
(255,255,255), 2)
    cv.line(draw_img, (x_mean, y_mean-30), (x_mean, y_mean+30),
(255,255,255), 2)
    return draw_img

#Main function
```

```
def main(img):
    gray = color2gray(img)
    lookup, mini, maxi = lookup_gen(gray)
    pseudocolored = convert(gray, lookup)
    final_img = draw(gray, pseudocolored, mini, maxi)
    return final_img

#Input filename
fname = input("Enter Filename: ")
img = cv.imread(fname)
cv.imshow('Original Image', img)
final = main(img)

#Display final image
cv.imshow('Pseudocolored Image', final)
cv.waitKey(0)
cv.destroyAllWindows

#Save final image
name = fname.split('.')
n_name = name[0]+'-color.'+name[1]
cv.imwrite(n_name, final)
```

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