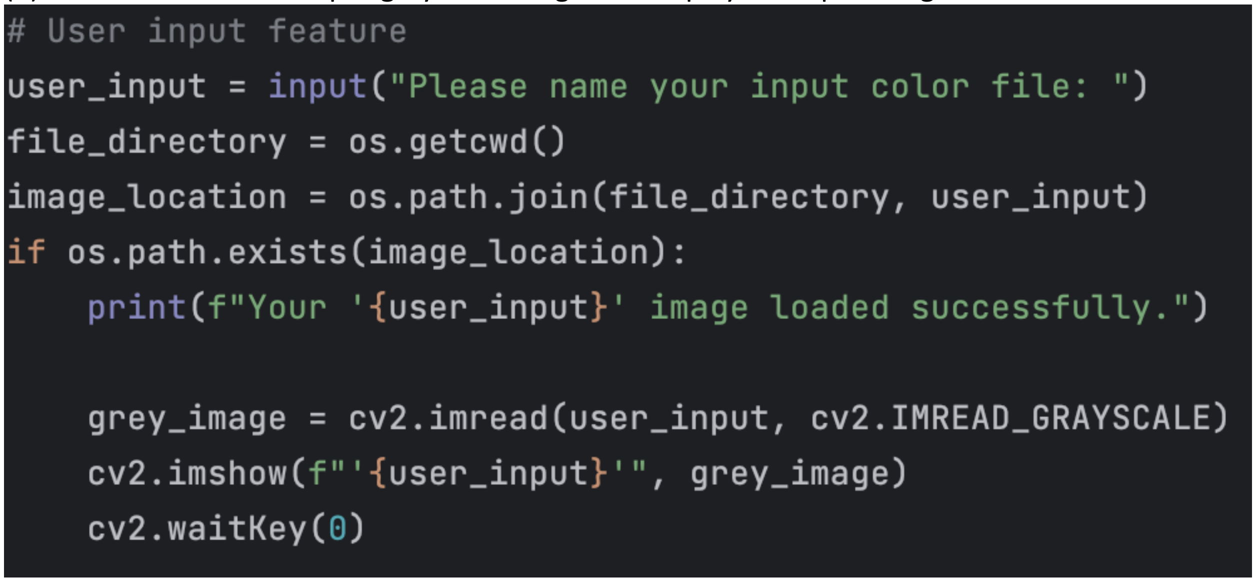
24-678: Computer Vision for Engineers
Ryan Wu
ID: weihuanw
PS2 Report
Due: Sep 22 2023

This file contains the following:

- night-vision-color.png
- thermal-color.png
- x-ray-color.png
- topography-color.png
- cracks-color.png
- readme.txt
- source code file(s) (attached to the end)

PS2-1 Converting a Greyscale Image to a Pseudo-color Image

(1) Ask the user for an input grayscale image and display the input image in the first window.



```python
# User input feature
user_input = input("Please name your input color file: ")
file_directory = os.getcwd()
image_location = os.path.join(file_directory, user_input)
if os.path.exists(image_location):
    print(f"Your '{user_input}' image loaded successfully.")

    grey_image = cv2.imread(user_input, cv2.IMREAD_GRAYSCALE)
    cv2.imshow(f"'{user_input}'", grey_image)
    cv2.waitKey(0)
```

Figure 1. Code used for asking user input and loading the image.

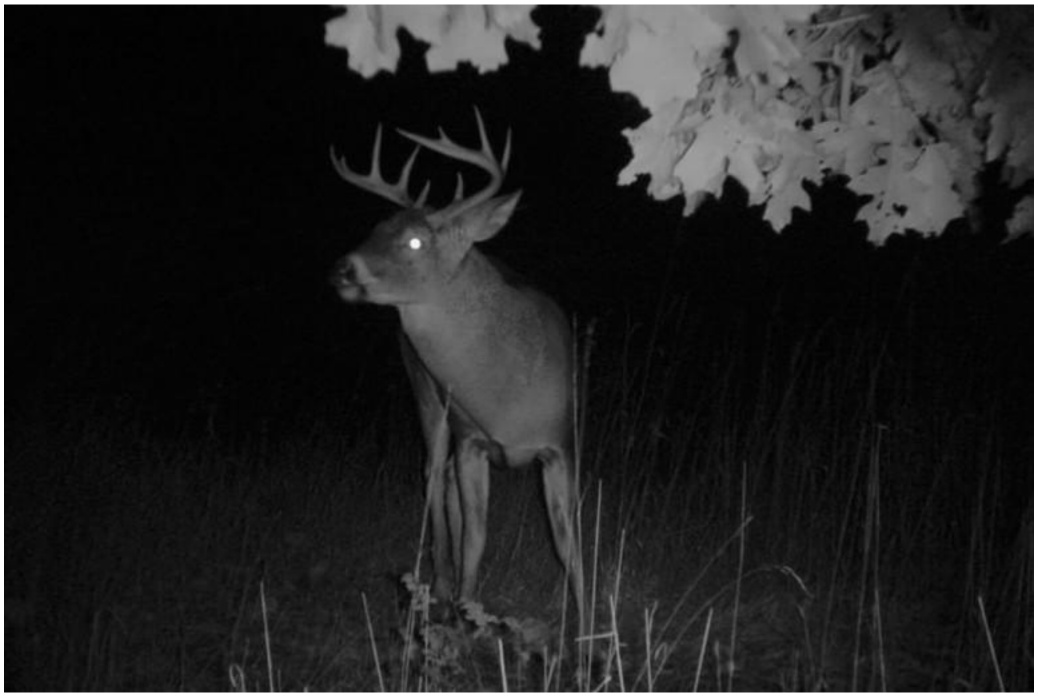


Figure 2. Night vision grayscale image.

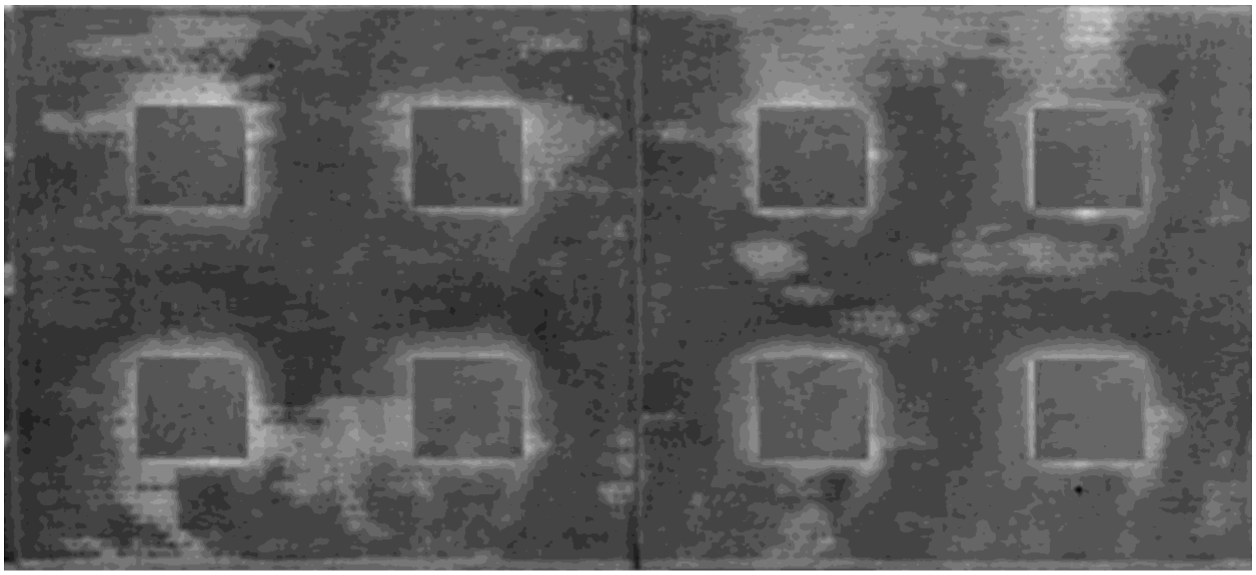


Figure 3. Thermal color grayscale image.

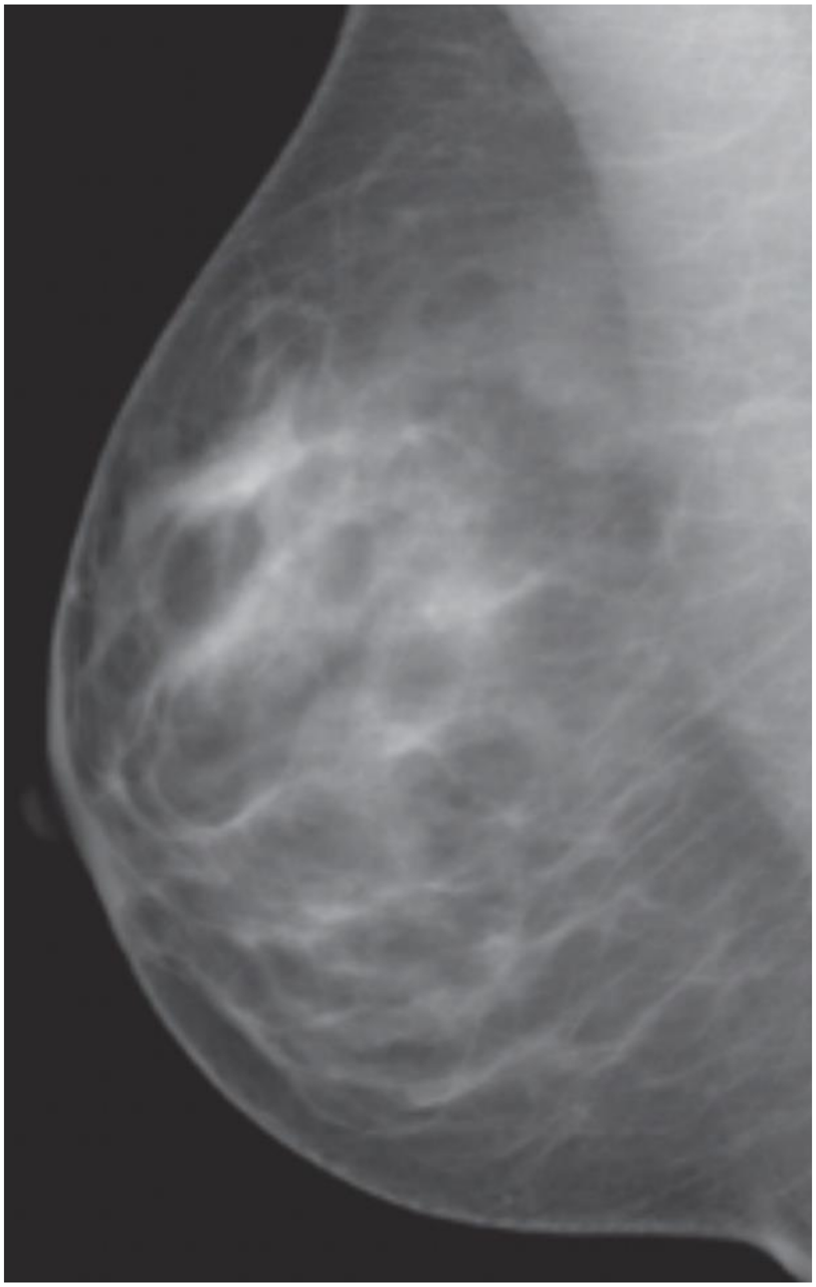


Figure 4. X-ray grayscale image.

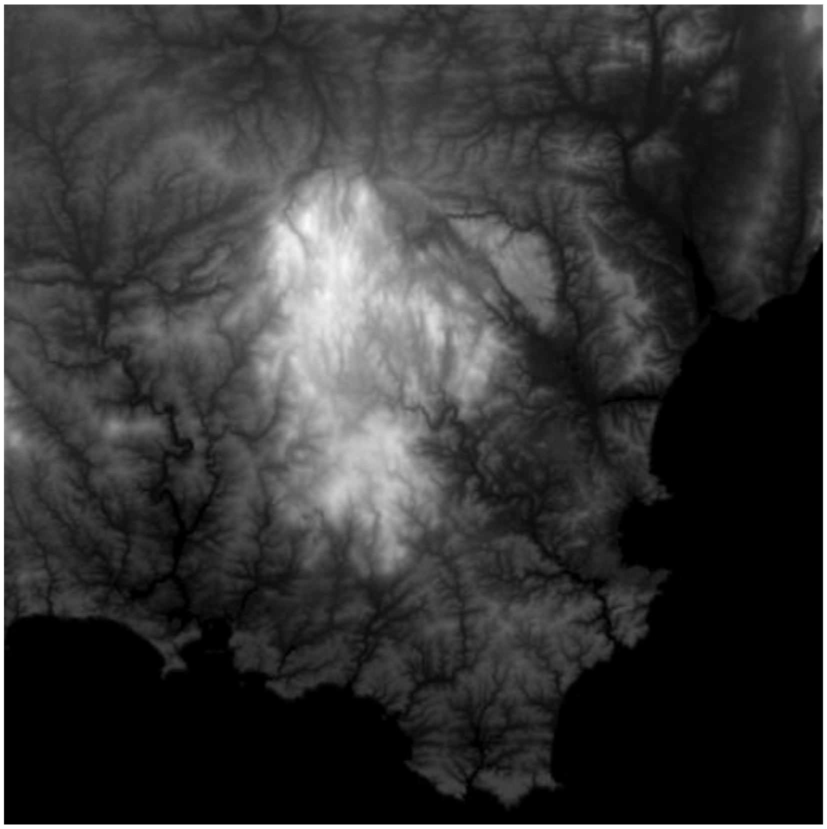


Figure 5. Topography grayscale image.

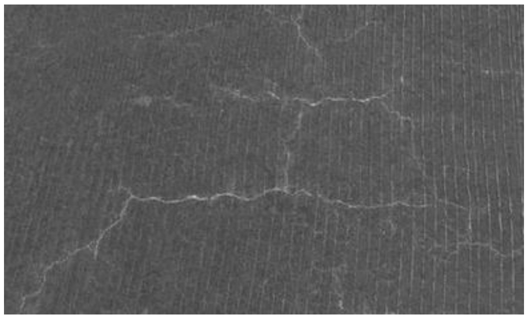
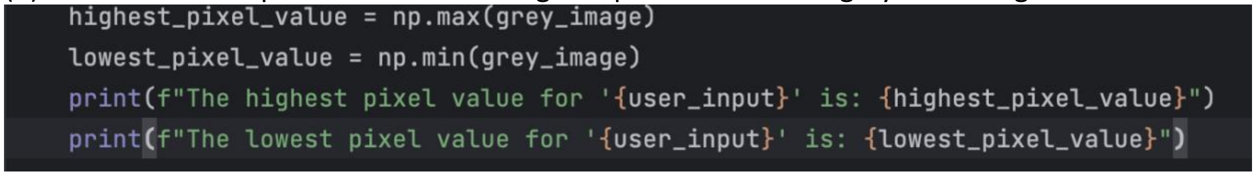


Figure 6. Cracks grayscale image.

(2) Find the lowest pixel value and the highest pixel value in the grayscale image.



```
    highest_pixel_value = np.max(grey_image)
    lowest_pixel_value = np.min(grey_image)
    print(f"The highest pixel value for '{user_input}' is: {highest_pixel_value}")
    print(f"The lowest pixel value for '{user_input}' is: {lowest_pixel_value}")
```

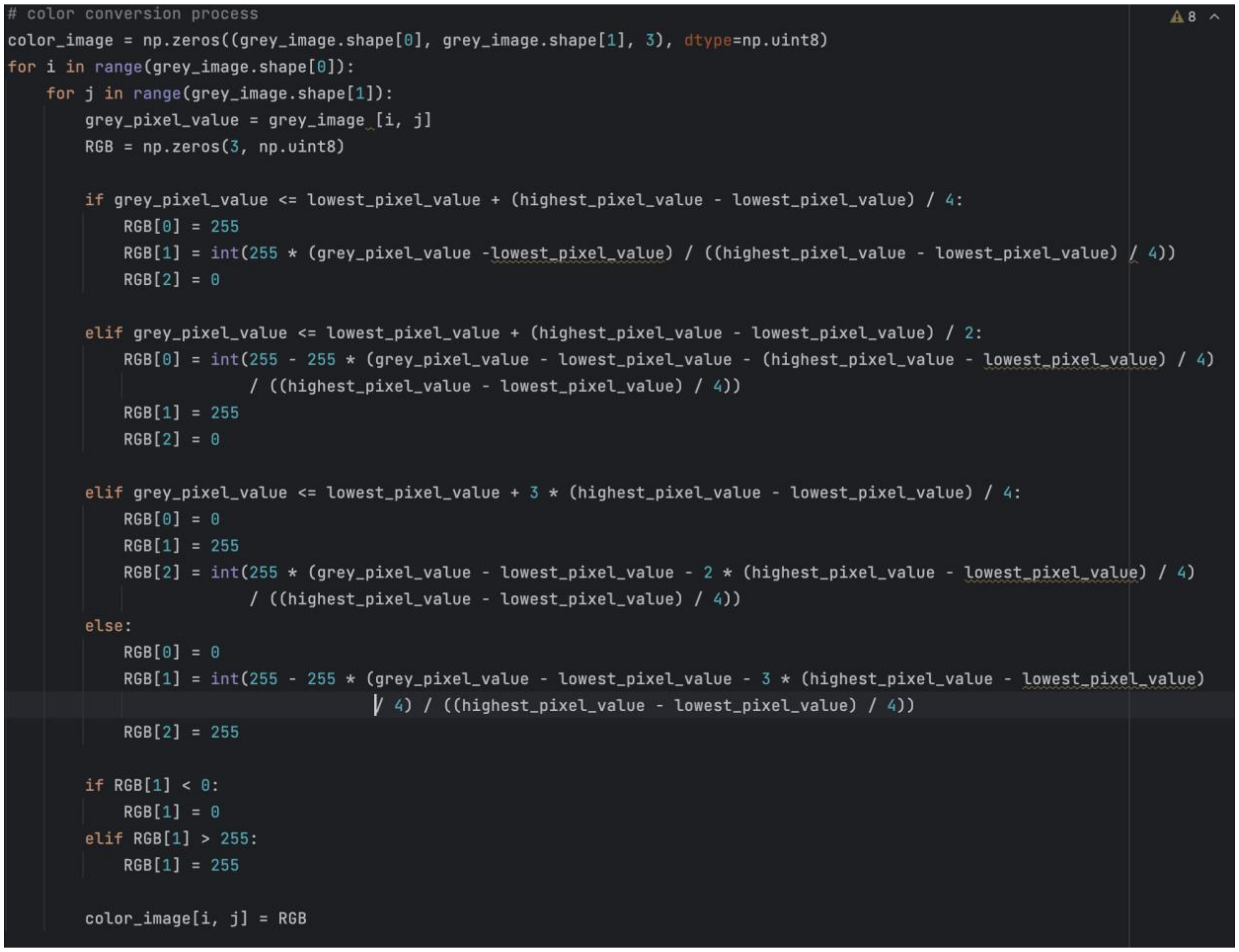
Figure 7. Code used to find the highest and lowest pixel values.

Table 1. Highest and lowest pixel values of each image.

Image name	**Highest Pixel Value**	**Lowest Pixel Value**
night-vision.png	248	0
thermal.png	207	8

x-ray.png	203	38
topography.png	252	0
cracks.png	203	53

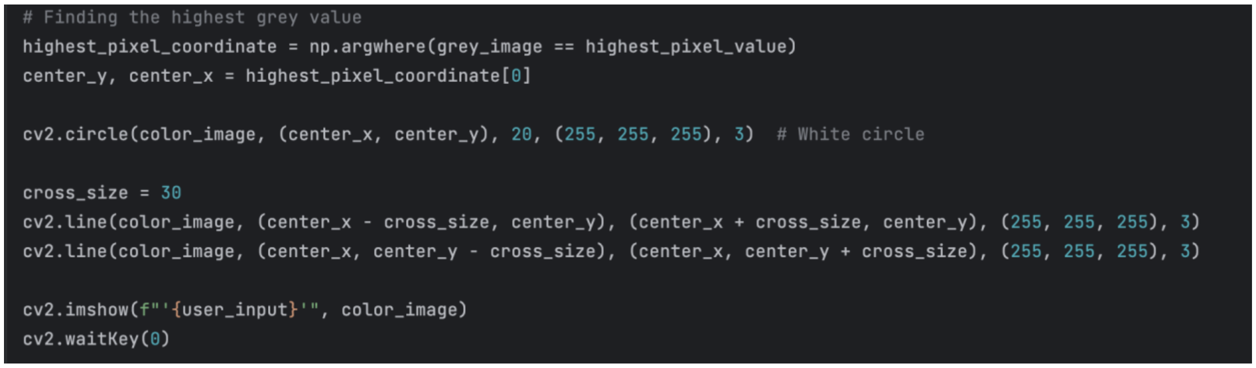
(3) Make a look-up table to convert the lowest gray value to blue and the highest gray value to red. The other gray values should be mapped to rainbow colors by the method explained in the lecture.



```
# color conversion process
color_image = np.zeros((grey_image.shape[0], grey_image.shape[1], 3), dtype=np.uint8)
for i in range(grey_image.shape[0]):
    for j in range(grey_image.shape[1]):
        grey_pixel_value = grey_image [i, j]
        RGB = np.zeros(3, np.uint8)

        if grey_pixel_value <= lowest_pixel_value + (highest_pixel_value - lowest_pixel_value) / 4:
            RGB[0] = 255
            RGB[1] = int(255 * (grey_pixel_value -lowest_pixel_value) / ((highest_pixel_value - lowest_pixel_value) / 4))
            RGB[2] = 0

        elif grey_pixel_value <= lowest_pixel_value + (highest_pixel_value - lowest_pixel_value) / 2:
            RGB[0] = int(255 - 255 * (grey_pixel_value - lowest_pixel_value - (highest_pixel_value - lowest_pixel_value) / 4)
                         / ((highest_pixel_value - lowest_pixel_value) / 4))
            RGB[1] = 255
            RGB[2] = 0

        elif grey_pixel_value <= lowest_pixel_value + 3 * (highest_pixel_value - lowest_pixel_value) / 4:
            RGB[0] = 0
            RGB[1] = 255
            RGB[2] = int(255 * (grey_pixel_value - lowest_pixel_value - 2 * (highest_pixel_value - lowest_pixel_value) / 4)
                         / ((highest_pixel_value - lowest_pixel_value) / 4))
        else:
            RGB[0] = 0
            RGB[1] = int(255 - 255 * (grey_pixel_value - lowest_pixel_value - 3 * (highest_pixel_value - lowest_pixel_value)
                                      / 4) / ((highest_pixel_value - lowest_pixel_value) / 4))
            RGB[2] = 255

        if RGB[1] < 0:
            RGB[1] = 0
        elif RGB[1] > 255:
            RGB[1] = 255

        color_image[i, j] = RGB
```

Figure 8. Code used for LUT and pixel color conversion.

(4) Using OpenCV functions, draw a cross in a circle to indicate the pixel of the highest gray value. Draw the cross and circle with white. If multiple pixels share the same highest gray value, place the cross and circle at the center of gravity of these pixels.



```
# Finding the highest grey value
highest_pixel_coordinate = np.argwhere(grey_image == highest_pixel_value)
center_y, center_x = highest_pixel_coordinate[0]

cv2.circle(color_image, (center_x, center_y), 20, (255, 255, 255), 3)  # White circle

cross_size = 30
cv2.line(color_image, (center_x - cross_size, center_y), (center_x + cross_size, center_y), (255, 255, 255), 3)
cv2.line(color_image, (center_x, center_y - cross_size), (center_x, center_y + cross_size), (255, 255, 255), 3)

cv2.imshow(f"'{user_input}'", color_image)
cv2.waitKey(0)
```

Figure 9. Code used for color image output.

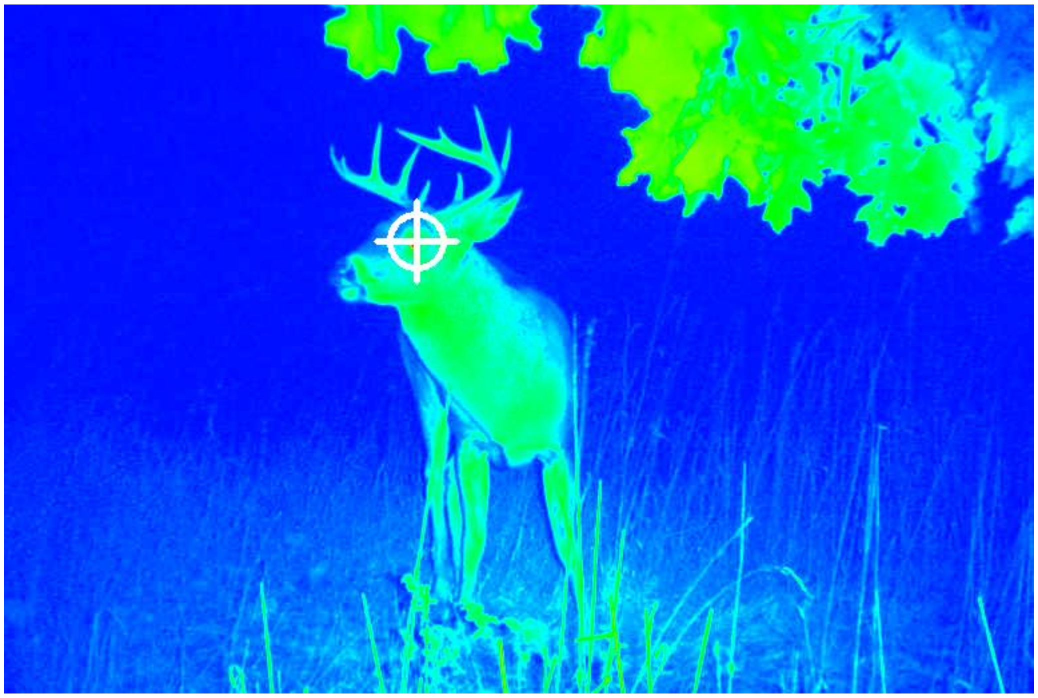


Figure 10. Night vision color image with the highest gray value labeled.

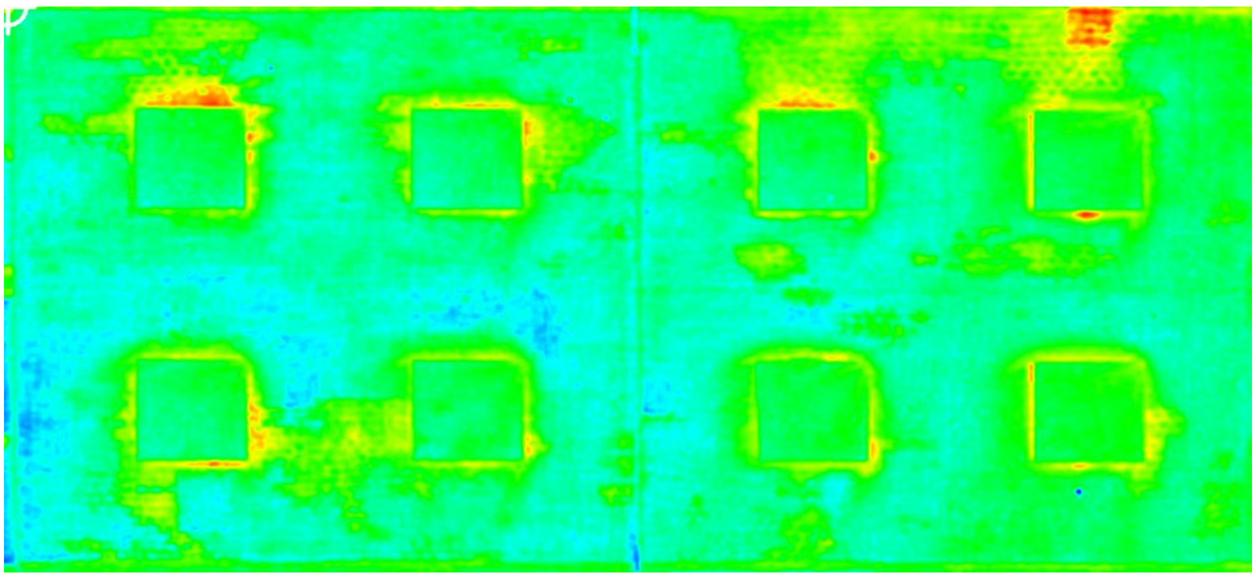


Figure 11. Thermal color image with the highest gray value labeled (top left corner).

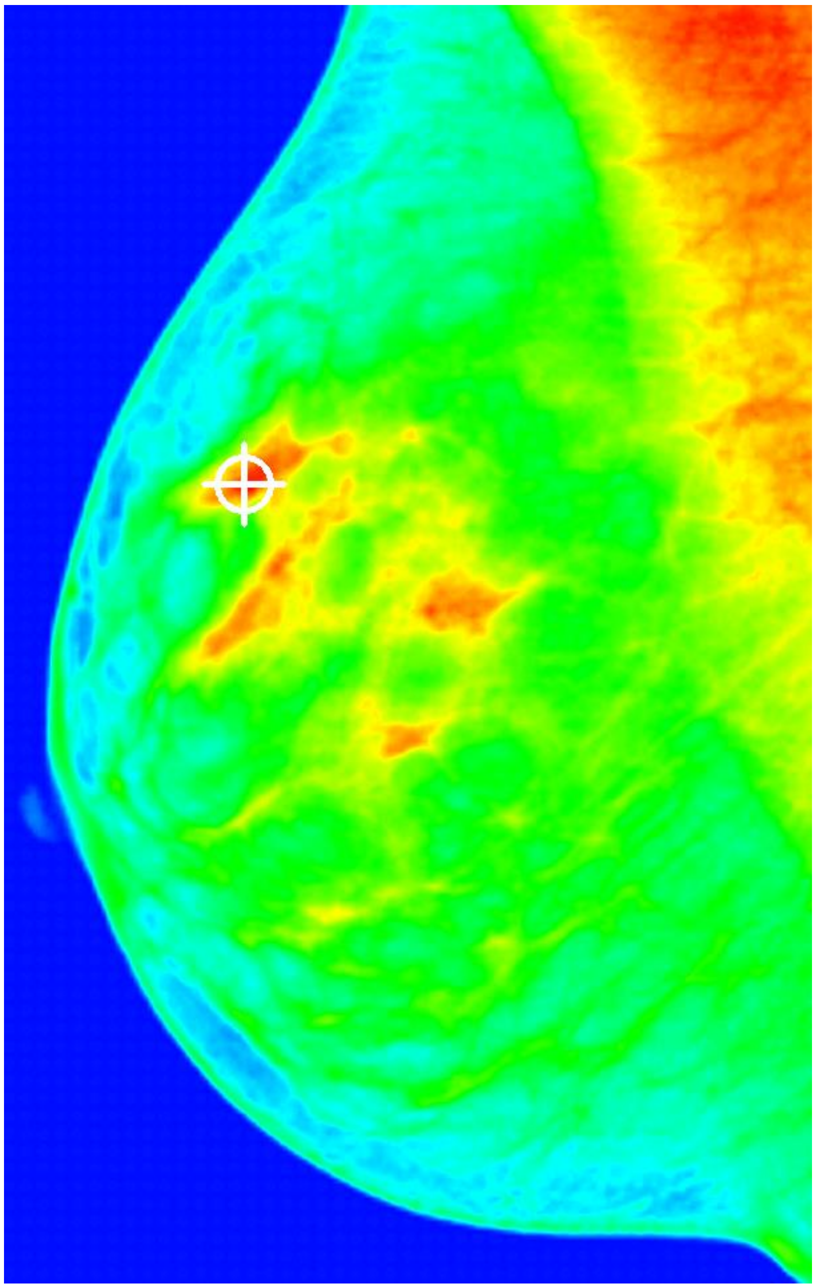


Figure 12. X-ray color image with the highest gray value labeled.

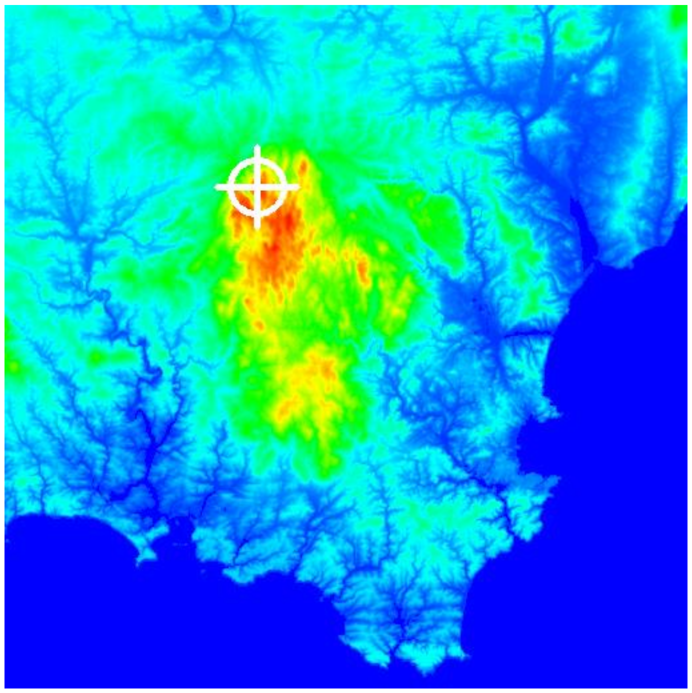


Figure 13. Topography color image with the highest gray value labeled.

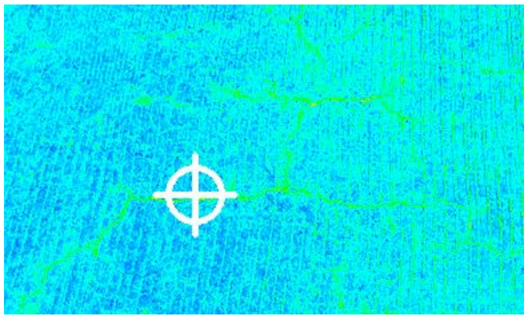
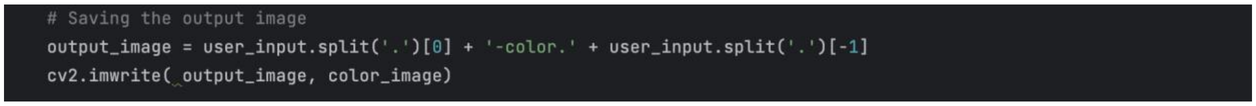


Figure 14. Cracks color image with the highest gray value labeled.

(5) Save the final color image as input-filename-color.png and display the file in the second window.


```
    # Saving the output image
    output_image = user_input.split('.')[0] + '-color.' + user_input.split('.')[-1]
    cv2.imwrite( output_image, color_image)
```
Figure 15. Code used for saving output images.

PS2 Read me text file

```
24-678: Computer Vision for Engineers
Ryan Wu
ID: weihuanw
PS2 Converting a grayscale image to a pseudo-color image

Operating system: macOS Ventura 13.5.2
IDE you used to write and run your code: PyCharm 2023.1.4 (Community
Edition)
The number of hours you spent to finish this problem: 12 hours.
```

```python

# (1)  Ask the user for an input grayscale image and display the input image in the first window.
# (2)  Find the lowest pixel value and the highest pixel value in the grayscale image.
# (3)  Make a look-up table to convert the  lowest gray value to blue and  the  highest  gray value to red.
#      The other gray values should be mapped to rainbow colors by the method explained in the lecture.
# (4)  Using  OpenCV  functions,  draw  a cross  in  a circle  to  indicate the pixel  of the  highest  gray  value.
#      Draw the cross and circle with white. If multiple pixels share the same highest gray value, place the
#      cross and circle at the center of gravity of these pixels.  Figure 4 shows a sample input image and
#      output image.
# (5)  Save the final color image as input-filename-color.png and display the file in the second window.
import cv2
import numpy as np
import os

# User input feature
user_input = input("Please name your input color file: ")
file_directory = os.getcwd()
image_location = os.path.join(file_directory, user_input)
if os.path.exists(image_location):
    print(f"Your '{user_input}' image loaded successfully.")

    grey_image = cv2.imread(user_input, cv2.IMREAD_GRAYSCALE)
    cv2.imshow(f"'{user_input}'", grey_image)
    cv2.waitKey(0)

    highest_pixel_value = np.max(grey_image)
    lowest_pixel_value = np.min(grey_image)

    # color conversion process
    color_image = np.zeros((grey_image.shape[0], grey_image.shape[1], 3), dtype=np.uint8)
    for i in range(grey_image.shape[0]):
        for j in range(grey_image.shape[1]):
            grey_pixel_value = grey_image [i, j]
            RGB = np.zeros(3, np.uint8)

            if grey_pixel_value <= lowest_pixel_value + (highest_pixel_value - lowest_pixel_value) / 4:
                RGB[0] = 255
                RGB[1] = int(255 * ( grey_pixel_value -lowest_pixel_value) / (( highest_pixel_value - lowest_pixel_value) / 4))
                RGB[2] = 0

            elif grey_pixel_value <= lowest_pixel_value + (highest_pixel_value - lowest_pixel_value) / 2:
                RGB[0] = int(255 - 255 * ( grey_pixel_value - lowest_pixel_value - ( highest_pixel_value - lowest_pixel_value) / 4) / (( highest_pixel_value - lowest_pixel_value) / 4))
                RGB[1] = 255
                RGB[2] = 0

            elif grey_pixel_value <= lowest_pixel_value + 3 * (highest_pixel_value - lowest_pixel_value) / 4:
                RGB[0] = 0
                RGB[1] = 255
                RGB[2] = int(255 * ( grey_pixel_value - lowest_pixel_value - 2 * ( highest_pixel_value - lowest_pixel_value) / 4) / (( highest_pixel_value - lowest_pixel_value) / 4))
            else:
                RGB[0] = 0
                RGB[1] = int(255 - 255 * (grey_pixel_value - lowest_pixel_value - 3 * (highest_pixel_value - lowest_pixel_value) / 4) / ((highest_pixel_value - lowest_pixel_value) / 4))
                RGB[2] = 255

            if RGB[1] < 0:
                RGB[1] = 0
            elif RGB[1] > 255:
                RGB[1] = 255

            color_image[i, j] = RGB

    # Finding the highest grey value
    highest_pixel_coordinate = np.argwhere(grey_image == highest_pixel_value)
    center_y, center_x = highest_pixel_coordinate[0]

    cv2.circle(color_image, (center_x, center_y), 20, (255, 255, 255), 3)  # White circle

    cross_size = 30
    cv2.line(color_image, (center_x - cross_size, center_y), (center_x + cross_size, center_y), (255, 255, 255), 3)
    cv2.line(color_image, (center_x, center_y - cross_size), (center_x, center_y + cross_size), (255, 255, 255), 3)

    cv2.imshow(f"'{user_input}'", color_image)
    cv2.waitKey(0)

    # Saving the output image
    output_image = user_input.split('.')[0] + '-color.' + user_input.split('.')[-1]
    cv2.imwrite( output_image, color_image)

    print(f"The highest pixel value for '{
```

```python
user_input}' is: {highest_pixel_value}")
    print(f"The lowest pixel value for '{user_input}' is: {lowest_pixel_value}")

else:
    print(f"Error: unable to load your input image.\nPlease make sure '{user_input}' is in the correct directory.")
    exit()


```