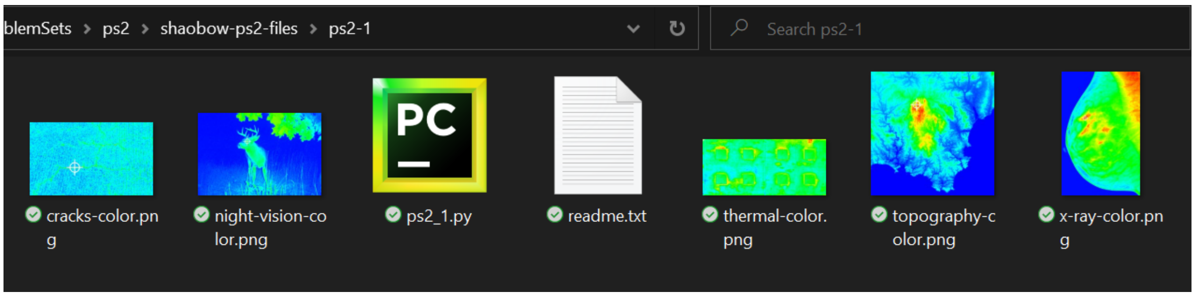
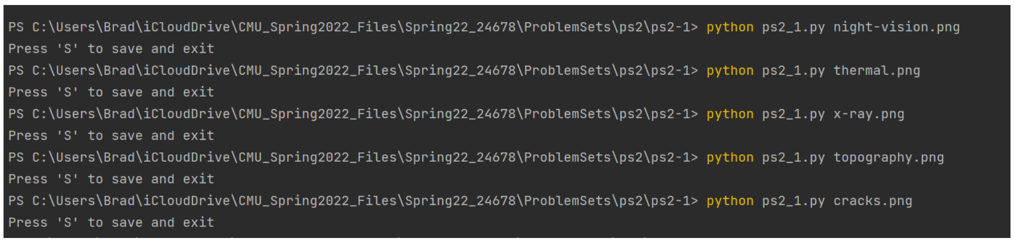
PS2-1Converting a grayscale image to a pseudo-color image

Shaobo Wang

Screenshots of all the files in the “ps2-1” folder:



Terminal printouts:

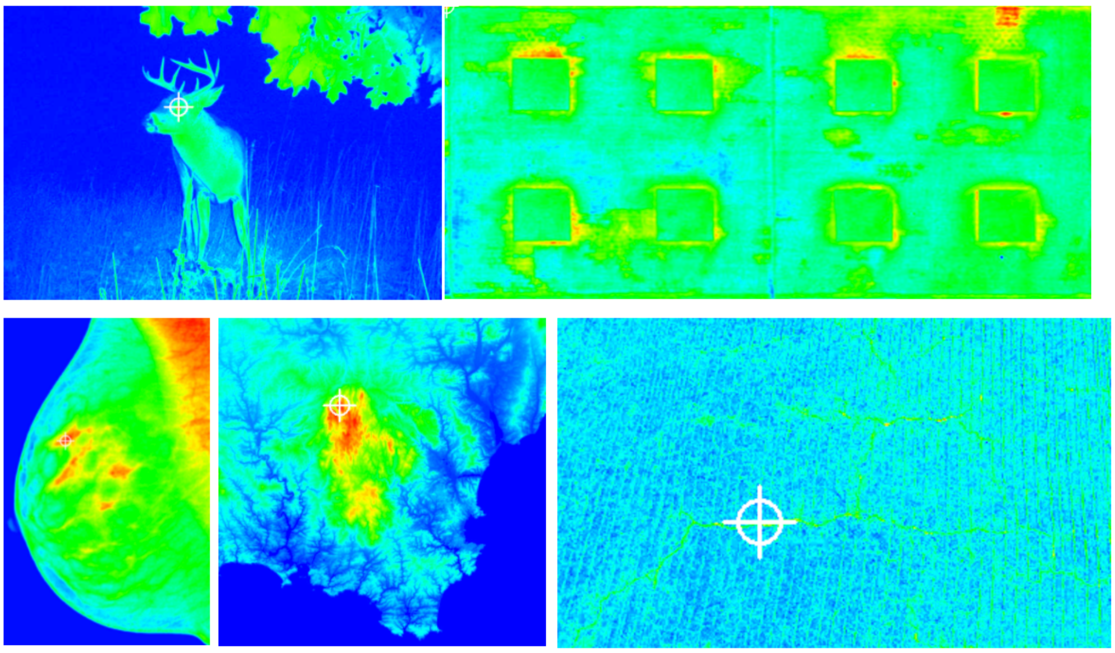


```
PS C:\Users\Brad\iCloudDrive\CMU_Spring2022_Files\Spring22_24678\ProblemSets\ps2\ps2-1> python ps2_1.py night-vision.png
Press 'S' to save and exit
PS C:\Users\Brad\iCloudDrive\CMU_Spring2022_Files\Spring22_24678\ProblemSets\ps2\ps2-1> python ps2_1.py thermal.png
Press 'S' to save and exit
PS C:\Users\Brad\iCloudDrive\CMU_Spring2022_Files\Spring22_24678\ProblemSets\ps2\ps2-1> python ps2_1.py x-ray.png
Press 'S' to save and exit
PS C:\Users\Brad\iCloudDrive\CMU_Spring2022_Files\Spring22_24678\ProblemSets\ps2\ps2-1> python ps2_1.py topography.png
Press 'S' to save and exit
PS C:\Users\Brad\iCloudDrive\CMU_Spring2022_Files\Spring22_24678\ProblemSets\ps2\ps2-1> python ps2_1.py cracks.png
Press 'S' to save and exit
```

Terminal commands:

```
>> python ps2_1.py night-vision.png
>> python ps2_1.py thermal.png
>> python ps2_1.py x-ray.png
>> python ps2_1.py topography.png
>> python ps2_1.py cracks.png
```

Output images:



Comments:

By indicating only the highest gray value directly, there could be cases where the outliers jeopardize the final result, such as in “thermal-color.png”.

Source code:

Operating system: Windows 10
IDE: PyCharm

```
import cv2
import sys
import numpy as np

MAXVALUE = 255
K = 100  # scale factor for trackbar
WHITE = (MAXVALUE, MAXVALUE, MAXVALUE)  # BGR white color

win_original_name = "Original Image"
win_colored_name = "Pseudo-color Image"


# insert string before
def insert_name(name, str2add):
    dot_idx = name.find(".")
    new_name = name[:dot_idx] + str2add + name[dot_idx:]
    return new_name


# pseudo color
def pseudo_color(img_in):
    # color mapping
    lut = np.zeros((MAXVALUE + 1, 1, 3), dtype=np.uint8)  # init look-up table
    max_gray = np.max(img_in)  # find maximum gray value
    min_gray = np.min(img_in)  # find minimum gray value
    interval = max_gray - min_gray  # gray value span
    for i in range(min_gray, max_gray + 1):  # map color wrt rainbow
        if i <= min_gray + interval/4:
            lut[i, 0, 0] = MAXVALUE  # blue
            lut[i, 0, 1] = MAXVALUE*4/interval*(i-min_gray)  # green
            lut[i, 0, 2] = 0  # red
        elif i <= min_gray + interval/2:
            lut[i, 0, 0] = MAXVALUE-MAXVALUE*4/interval*(i-(min_gray + 
interval/4))
            lut[i, 0, 1] = MAXVALUE
            lut[i, 0, 2] = 0
        elif i <= min_gray + 3*interval/4:
            lut[i, 0, 0] = 0
            lut[i, 0, 1] = MAXVALUE
            lut[i, 0, 2] = MAXVALUE*4/interval*(i-(min_gray + interval/2))
        else:
            lut[i, 0, 0] = 0
            lut[i, 0, 1] = MAXVALUE-MAXVALUE*4/interval*(i-(min_gray + 
3*interval/4))
            lut[i, 0, 2] = MAXVALUE
    img_out = cv2.LUT(img_in, lut)  # look-up table transform
    # indicate highest gray value
    max_gray_row = np.argwhere(img_in == max_gray)[:, [0]]  # row indices of max
gray pts
    max_gray_col = np.argwhere(img_in == max_gray)[:, [1]]  # column indices of
max gray pts
    com_row = int(np.average(max_gray_row))  # row index COM of max gray pts
    com_col = int(np.average(max_gray_col))  # column index COM of max gray pts
    half_length = 25  # half of line length
    radius = int(0.6*half_length)  # circle radius
    c = (com_col, com_row)  # center pt
    s1 = (com_col-half_length, com_row)  # start pt1
    e1 = (com_col+half_length, com_row)  # end pt1
    s2 = (com_col, com_row-half_length)  # start pt2
    e2 = (com_col, com_row+half_length)  # end pt2
    cv2.line(img_out, s1, e1, WHITE, 2)  # horizontal line
    cv2.line(img_out, s2, e2, WHITE, 2)  # vertical line
    cv2.circle(img_out, c, radius, WHITE, 2)  # circle
    return img_out


if __name__ == "__main__":
    # get input arguments
    args = sys.argv
    assert (len(args) == 2)  # make sure two arguments input
    img_name = args[1]  # input image path

    cv2.namedWindow(win_original_name)
    cv2.namedWindow(win_colored_name)

    # read original image
    img = cv2.imread(img_name)
    if img is None:
        sys.exit("Could not read the image.")
    cv2.imshow(win_original_name, img)

    # img = np.zeros((512, 512, 3), dtype=np.uint8)
    img_output = pseudo_color(img)
    cv2.imshow(win_colored_name, img_output)

    print("Press 'S' to save and exit")
    key = cv2.waitKey(-1)

    if key == ord("s"):
        #  press 'S' to save
        img_output_name = insert_name(img_name, "-color")
        cv2.imwrite(img_output_name, img_output)
        cv2.destroyAllWindows()

    elif key == ord("q") or key == ord("x") or key == 27:
        cv2.destroyAllWindows()
        # press 'q', 'x', 'ESC' to quit
```