24-678: Computer Vision for Engineers
Ryan Wu
ID: weihuanw
PS1 Report
Due: Sep 16 2023

This file contains the following:
PS1-2 Read color images, apply thresholding, and change colors

- grayscale image files: “circuit_grayscale.png” and “crack_grayscale.png”
- binary image files: “circuit_binary.png” and “crack_binary.png”
- output image files: “circuit_output.png” and “crack_output.png”
- readme.txt
- source code file(s) (attached to the end)

PS1-3 Gamma correction

- gamma-corrected images: “smiley_gcorrected.jpg” and “carnival_gcorrected.jpg”
- readme.txt includes
- source code file(s) (attached to the end)

**PS1-2 Grayscale**

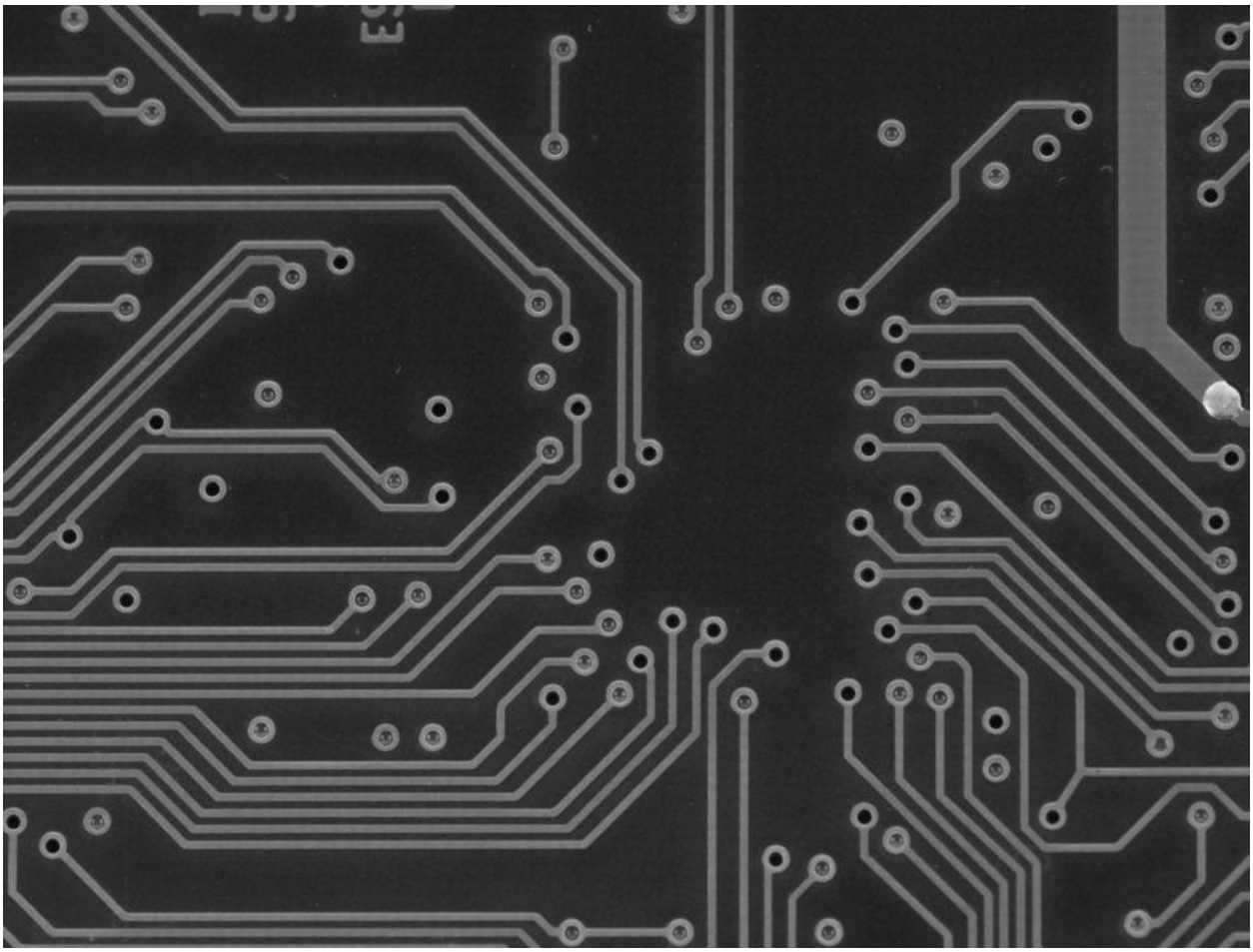

*Figure 1. Circuit grayscale image.*

**PS1-2 Grayscale**

*Figure 2. Crack grayscale image.*

**PS1-2 Binary**

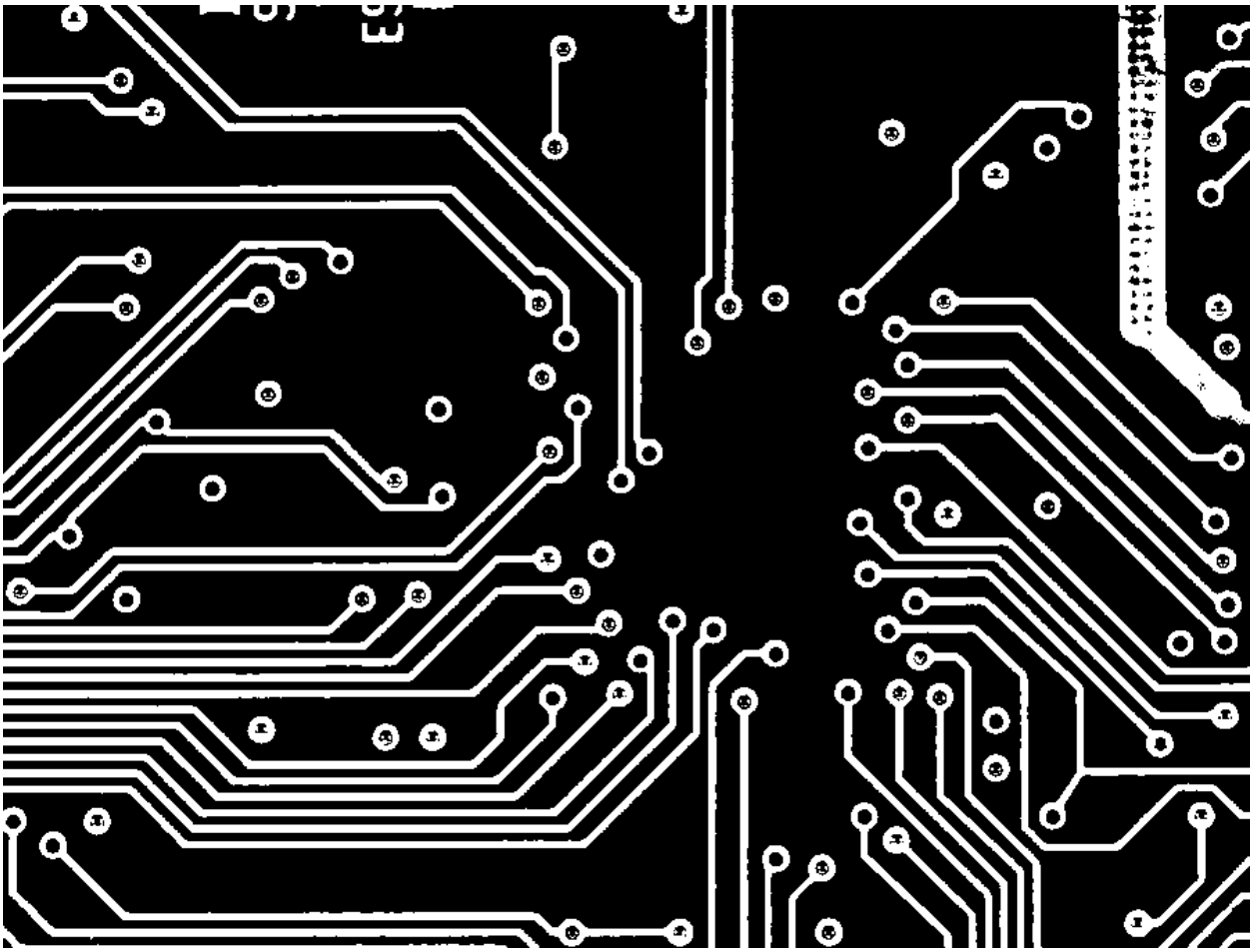

*Figure 3. Circuit binary image.*

**PS1-2 Binary**

*Figure 4. Crack binary image.*

**PS1-2 Output**

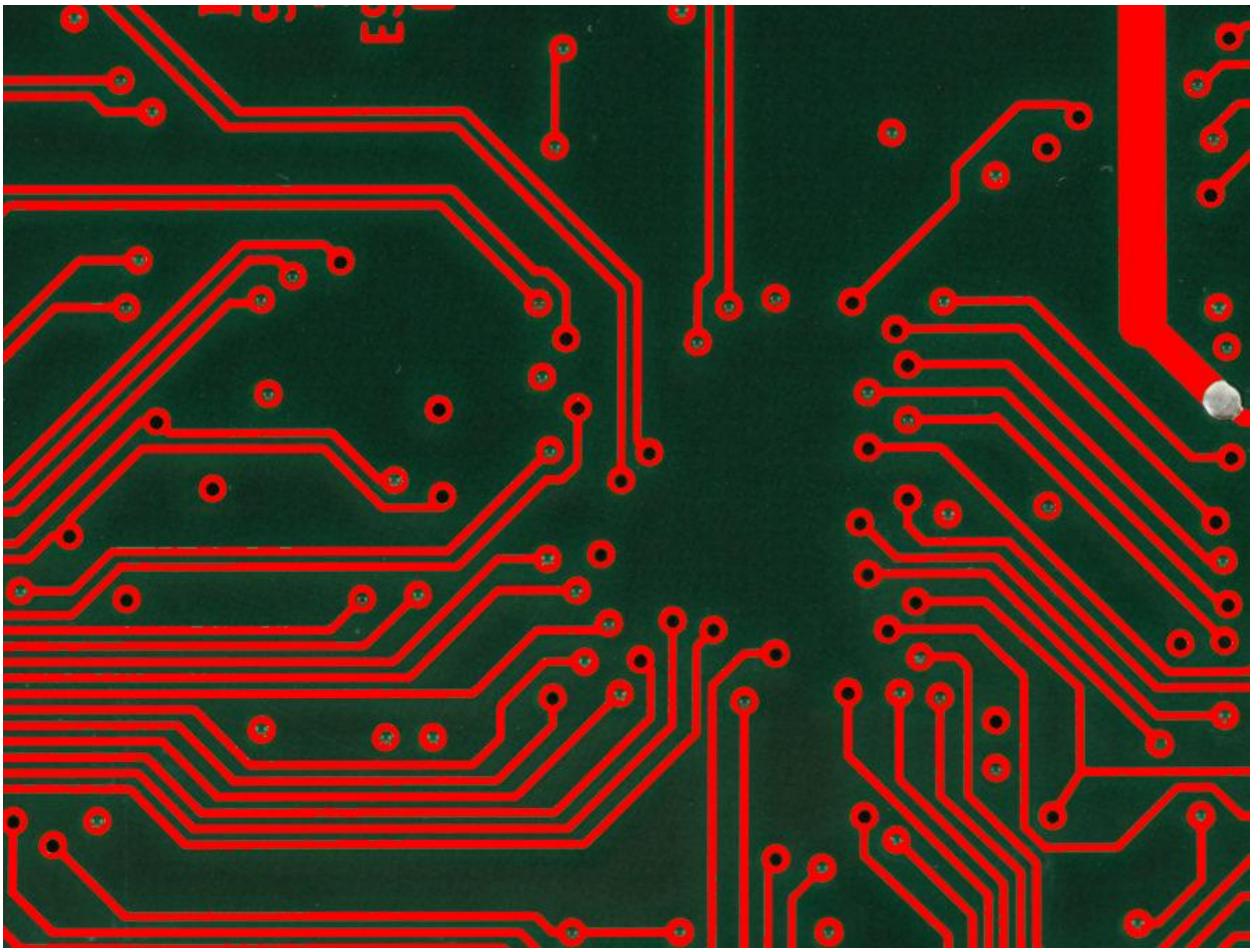

*Figure 5. Circuit output color image.*

**PS1-2 Output**

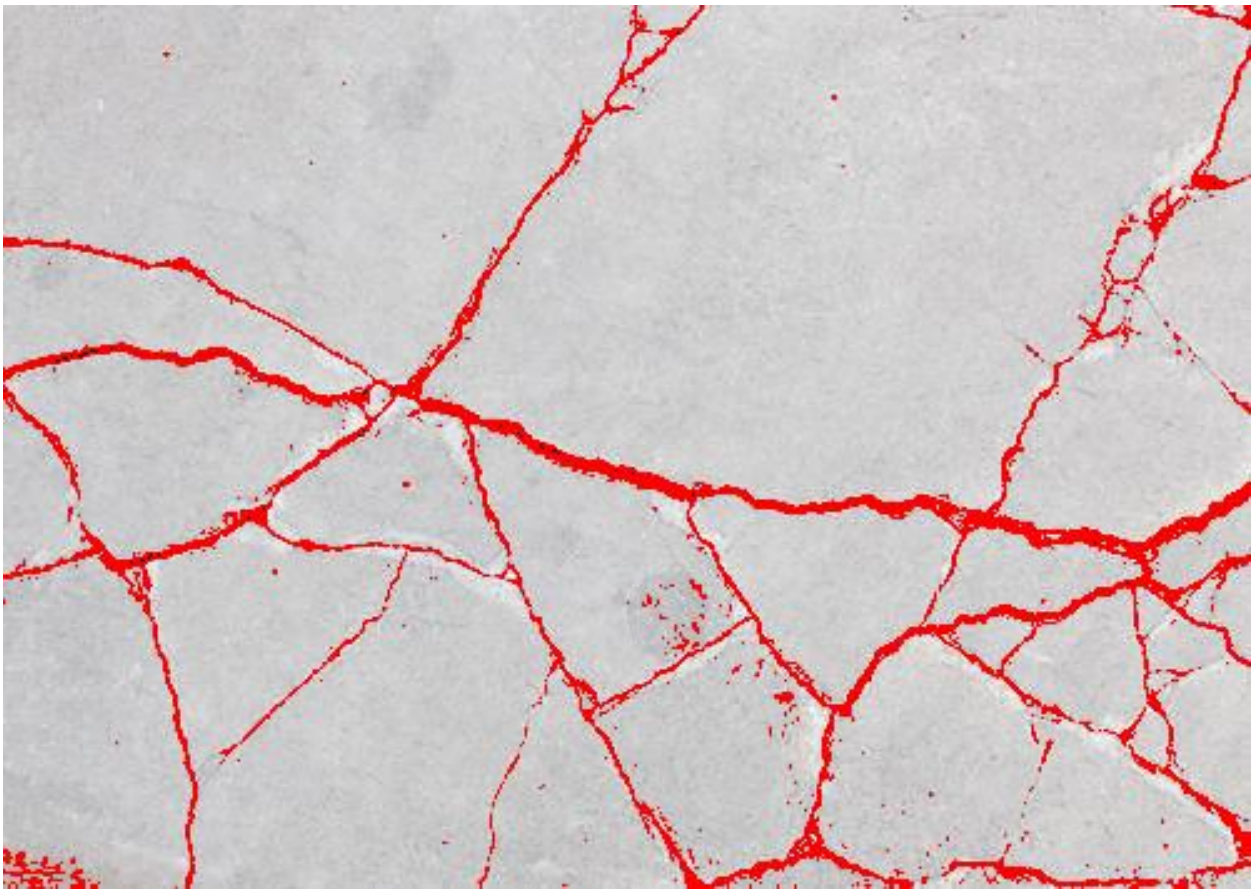

*Figure 6. Crack output color image.*

**PS1-2 Read me text file**

```
24-678: Computer Vision for Engineers
Ryan Wu
ID: weihuanw
PS1-2 Read color images, apply thresholding, and change colors

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.
```

**PS1-3 Gamma correction**

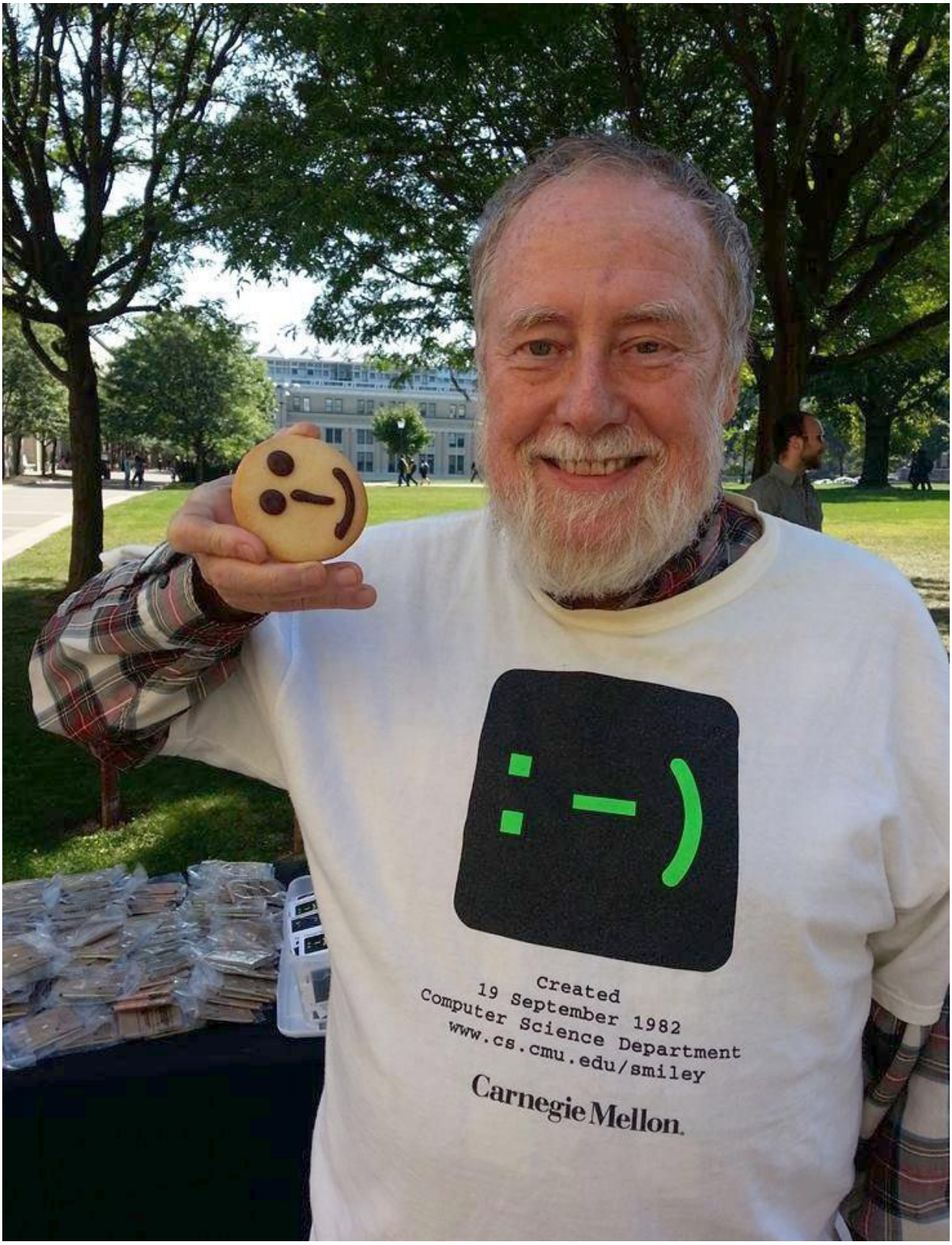


*Figure 7. Smiley with gamma correction.*

**PS1-3 Gamma correction**


*Figure 8. Carnival with gamma correction.*

**PS1-3 Read me text file**

```
24-678: Computer Vision for Engineers
Ryan Wu
ID: weihuanw
PS1-3 Gamma correction

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: 5 hours.
```

```python
import cv2
import numpy as np
import os

# PS1-2 (1) 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.")
else:
    print(f"Error: unable to load your input image. \nPlease make sure '{user_input}' is in the correct directory.")
    exit()

emphasize_choice = input("Please indicate your emphasize region, 'dark' or 'bright'?: ")
if emphasize_choice == 'dark':
    print(f"You choose '{emphasize_choice}' as your emphasize region .")
elif emphasize_choice == 'bright':
    print(f"You choose '{emphasize_choice}' as your emphasize region .")
else:
    print("Error: unable execute region emphasis.")
    exit()

# PS1-2 (1) Open and load image files
if user_input == 'circuit.png':
    circuit_color = cv2.imread(user_input)
    cv2.imshow('Circuit Color Image', circuit_color)
    cv2.waitKey(0)
    cv2.destroyWindow('Circuit Color Image')
elif user_input == 'crack.png':
    crack_color = cv2.imread(user_input)
    cv2.imshow('Crack Color Image', crack_color)
```

```
33      cv2.waitKey(0)
34      cv2.destroyWindow('Crack Color Image')
35
36  # PS1-2 (2) Convert color image to grayscale
37  if user_input == 'circuit.png':
38      circuit_color = cv2.imread(user_input)
39      circuit_grey = cv2.cvtColor(circuit_color, cv2.
    COLOR_BGR2GRAY)
40      cv2.imshow('Circuit Grey Image', circuit_grey)
41      cv2.waitKey(0)
42      cv2.destroyWindow('Circuit Grey Image')
43      cv2.imwrite('circuit_greyscale.png',
    circuit_grey)
44  elif user_input == 'crack.png':
45      circuit_color = cv2.imread(user_input)
46      crack_grey = cv2.cvtColor(circuit_color, cv2.
    COLOR_BGR2GRAY)
47      cv2.imshow('Crack Grey Image', crack_grey)
48      cv2.waitKey(0)
49      cv2.destroyWindow('Circuit Grey Image')
50      cv2.imwrite('crack_greyscale.png', crack_grey)
51
52  # PS1-2 (3) Convert greyscale image to binary
53  if user_input == 'circuit.png':
54      circuit_grey = cv2.imread('circuit_greyscale.
    png', cv2.IMREAD_GRAYSCALE)
55      _, circuit_binary = cv2.threshold(circuit_grey
    , 95, 255, cv2.THRESH_BINARY)
56      cv2.imshow('Circuit Binary Image',
    circuit_binary)
57      cv2.waitKey(0)
58      cv2.destroyWindow('Circuit Binary Image')
59      cv2.imwrite('circuit_binary.png',
    circuit_binary)
60  elif user_input == 'crack.png':
61      crack_grey = cv2.imread('crack_greyscale.png',
    cv2.IMREAD_GRAYSCALE)
62      _, crack_binary = cv2.threshold(crack_grey, 180
    , 255, cv2.THRESH_BINARY)
63      cv2.imshow('Crack Binary Image', crack_binary)
64      cv2.waitKey(0)
```

```
65      cv2.destroyWindow('Crack Binary Image')
66      cv2.imwrite('crack_binary.png', crack_binary)
67
68  # PS1-2 (4) Painting image pixel
69  # Circuit image color conversion
70  if user_input == 'circuit.png':
71      circuit_color = cv2.imread(user_input)
72      circuit_color_hsv = cv2.cvtColor(circuit_color
    , cv2.COLOR_BGR2HSV)
73      lower_green = np.array([10, 100, 100])
74      upper_green = np.array([20, 100, 100])
75      lower_yellow = np.array([10, 85, 100])
76      upper_yellow = np.array([70, 255, 255])
77
78      mask_green = cv2.inRange(circuit_color_hsv,
    lower_green, upper_green)
79      mask_yellow = cv2.inRange(circuit_color_hsv,
    lower_yellow, upper_yellow)
80      mask_green_yellow_circuit = cv2.bitwise_or(
    mask_green, mask_yellow)
81      mask_red_circuit = np.zeros_like(
    mask_green_yellow_circuit)
82      if emphasize_choice == 'bright':
83          mask_red_circuit[mask_green_yellow_circuit
     > 0] = 255
84          circuit_output_image_bright =
    circuit_color.copy()
85          circuit_output_image_bright[np.where(
    mask_red_circuit == 255)] = [0, 0, 255]
86
87          cv2.imshow('Circuit Output Image (Bright)'
    , circuit_output_image_bright)
88          cv2.imwrite('circuit_output.png',
    circuit_output_image_bright)
89          cv2.waitKey(0)
90          cv2.destroyWindow('Circuit Output Image (
    Bright)')
91      elif emphasize_choice == 'dark':
92          mask_red_circuit[mask_green_yellow_circuit
     > 0] = 255
93          circuit_output_image_dark = circuit_color.
```

```
 93 copy()
 94         circuit_output_image_dark[np.where(
    mask_red_circuit != 255)] = [0, 0, 255]
 95 
 96         cv2.imshow('Circuit Output Image (Dark)',
    circuit_output_image_dark)
 97         cv2.imwrite('circuit_output.png',
    circuit_output_image_dark)
 98         cv2.waitKey(0)
 99         cv2.destroyWindow('Circuit Output Image (
    Dark)')
100 
101 # Crack image color conversion
102 elif user_input == 'crack.png':
103     crack_color = cv2.imread(user_input)
104     crack_color_hsv = cv2.cvtColor(crack_color,
    cv2.COLOR_BGR2HSV)
105     lower_black = np.array([0, 0, 0])
106     upper_black = np.array([180, 180, 180])
107 
108     mask_black = cv2.inRange(crack_color_hsv,
    lower_black, upper_black)
109     mask_black_crack = cv2.bitwise_or(mask_black,
    mask_black)
110     mask_red_crack = np.zeros_like(
    mask_black_crack)
111     if emphasize_choice == 'bright':
112         mask_red_crack[mask_black_crack > 0] = 255
113         crack_output_image_bright = crack_color.
    copy()
114         crack_output_image_bright[np.where(
    mask_black_crack != 255)] = [0, 0, 255]
115 
116         cv2.imshow('Crack Output Image (Bright)',
    crack_output_image_bright)
117         cv2.imwrite('crack_output.png',
    crack_output_image_bright)
118         cv2.waitKey(0)
119         cv2.destroyWindow('Crack Output Image (
    Bright)')
120     elif emphasize_choice == 'dark':
```

```
121         mask_black_crack[mask_black_crack > 0] =
    255
122         crack_output_image_dark = crack_color.copy
    ()
123         crack_output_image_dark[np.where(
    mask_black_crack == 255)] = [0, 0, 255]
124
125         cv2.imshow('Crack Output Image (Dark)',
    crack_output_image_dark)
126         cv2.imwrite('crack_output.png',
    crack_output_image_dark)
127         cv2.waitKey(0)
128         cv2.destroyWindow('Crack Output Image (
    Dark)')
129 exit()
```

```python
import cv2
import numpy as np
import os

# PS1-3 (3) Gamma correction Function
def gamma_correction(image_input, gamma=1.0):
    image_input = cv2.imread(user_input)
    gamma_table = np.array([((i/255) ** gamma) * 255 for i in np.arange(0, 256)]).astype("uint8")
    output_image = cv2.LUT(image_input, gamma_table)
    return output_image

# PS1-3 (1) User input file
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.")
    image_input = cv2.imread(user_input)
    cv2.imshow('Input image', image_input)
else:
    print(f"Error: unable to load your input image. \nPlease make sure '{user_input}' is in the correct directory.")
    exit()

# PS1-3 (2) (4) User input gamma correction; Saving final output files
user_gamma_value = input("Please indicate your desire gamma correction value: ")
gamma_value = float(user_gamma_value)
if gamma_value > 0:
    print(f"You choose '{gamma_value}' as your gamma correction value.")
    gamma_corrected_image = gamma_correction(user_input, gamma=gamma_value)
    cv2.imshow('Output image after gamma correction
```

```
', gamma_corrected_image)
    final_image = user_input.split('.')[0] + '_gcorrected.' + user_input.split('.')[-1]
    cv2.imwrite( final_image, gamma_corrected_image)
    cv2.waitKey(0)
    cv2.destroyWindow('Circuit Color Image')
elif gamma_value < 0:
    print(f"'{gamma_value}' is an invalid gamma correction value.")
    exit()
else:
    print(f"'{gamma_value}' is an invalid gamma correction value.")
    exit()
exit()
```