```
1 from skimage.io import imread
 2 import matplotlib.pyplot as plt
 3 from skimage.filters import roberts
 4 from skimage.morphology import (
       dilation,
 5
 6
       square,
 7
       remove small objects,
       remove small holes,
 8
 9
10 import numpy as np
11 from pathlib import Path
12
13 ROOT = Path( file ).parent
14
15 def find_red_diffusion_boundary(img: np.ndarray) -> np.ndarray:
       """Find the red diffusion boundary.
16
17
       Args:
18
19
           img (np.ndarray): images.
20
21
       Returns:
22
           np.ndarray: binary images where 1s indicate the boundary.
23
24
       # Extract the red channel from the image and normalize it.
25
       img = img[:, :, 0] / 255.0
26
27
       # Convert the image to binary image with thresholding.
28
       img = img > 0.06
29
       # Connected neighboring pixels by filling small holes that are less than 64 pixels.
30
       img = remove_small_holes(img, 64)
31
32
       # Reduce noises by eliminating connected components that are less than 20,000 pixels.
33
       img = remove small objects(img, 20000)
34
35
36
       # Soften edges by dilating the image with 3x3 matrix
       img = dilation(img, square(3))
37
38
       # Fill in the remaining holes
39
       img = remove small holes(img, 128)
40
41
42
       # Use roberts filter to find edges
43
       img = roberts(img)
       img = img != 0
44
45
46
       return img
47
48
  def find_green_diffusion_boundary(img: np.ndarray) -> np.ndarray:
49
       """Find the green diffusion boundary.
50
51
52
53
           img (np.ndarray): images.
54
55
       Returns:
56
           np.ndarray: binary images where 1s indicate the boundary.
57
```

localhost:50030 1/2

```
# Extract the green channel from the image and normalize it.
 58
 59
        img = img[:, :, 1] / 255.0
60
        # Convert the image to binary image with thresholding.
61
62
        img = img > 0.06
63
        # Connected neighboring pixels by filling small holes that are less than 128 pixels.
64
        img = remove small holes(img, 128)
65
66
67
        # Reduce noises by eliminating connected components that are less than 64 pixels.
        img = remove small objects(img, 64)
68
69
 70
        # Soften edges by dilating the image with 4x4 matrix
        img = dilation(img, square(4))
 71
72
73
       # Fill in the remaining holes
74
        img = remove small holes(img, 1024)
75
76
        # Use roberts filter to find edges
77
        img = roberts(img)
78
        img = img != 0
79
80
        return img
81
82
 83 def problem 1(in img: str = ROOT/"dyes.png", out img: str =ROOT/"problem 1 result.png") ->
   None:
        """Find diffusion boundaries of the red and the green pixels.
84
85
86
        Args:
            in img (str, optional): path to the input image. Defaults to "./dyes.png".
87
            out img (str, optional): path to the output image. Defaults to
88
    "./problem1_reuslt.png".
89
90
        # Load the image
        img = imread(in img)
91
92
93
        # Find boundaries
        red diffusion boundary = find red diffusion boundary(img)
94
        green diffusion boundary = find green diffusion boundary(img)
95
96
97
       # Find boundaries' coordinates
98
        red coordinates = np.where(red diffusion boundary == 1)
99
        green coordinates = np.where(green diffusion boundary == 1)
100
        # Plot boundaries
101
        img[red coordinates[0], red coordinates[1]] = [255, 0, 0]
102
103
        img[green coordinates[0], green coordinates[1]] = [0, 255, 0]
104
105
        # Save the resulting image
106
        plt.imsave(out img, img)
107
108
109 if name == " main ":
110
       problem 1()
```

localhost:50030 2/2