

```
1 from skimage.io import imread
2 import matplotlib.pyplot as plt
3 from skimage.filters import roberts
4 from skimage.morphology import (
5     dilation,
6     square,
7     remove_small_objects,
8     remove_small_holes,
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:
16     """Find the red diffusion boundary.
17
18     Args:
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
30     # Connected neighboring pixels by filling small holes that are less than 64 pixels.
31     img = remove_small_holes(img, 64)
32
33     # Reduce noises by eliminating connected components that are less than 20,000 pixels.
34     img = remove_small_objects(img, 20000)
35
36     # Soften edges by dilating the image with 3x3 matrix
37     img = dilation(img, square(3))
38
39     # Fill in the remaining holes
40     img = remove_small_holes(img, 128)
41
42     # Use roberts filter to find edges
43     img = roberts(img)
44     img = img != 0
45
46     return img
47
48
49 def find_green_diffusion_boundary(img: np.ndarray) -> np.ndarray:
50     """Find the green diffusion boundary.
51
52     Args:
53         img (np.ndarray): images.
54
55     Returns:
56         np.ndarray: binary images where 1s indicate the boundary.
57     """
```

```
58 # Extract the green channel from the image and normalize it.
59 img = img[:, :, 1] / 255.0
60
61 # Convert the image to binary image with thresholding.
62 img = img > 0.06
63
64 # Connected neighboring pixels by filling small holes that are less than 128 pixels.
65 img = remove_small_holes(img, 128)
66
67 # Reduce noises by eliminating connected components that are less than 64 pixels.
68 img = remove_small_objects(img, 64)
69
70 # Soften edges by dilating the image with 4x4 matrix
71 img = dilation(img, square(4))
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:
84     """Find diffusion boundaries of the red and the green pixels.
85
86     Args:
87         in_img (str, optional): path to the input image. Defaults to "./dyes.png".
88         out_img (str, optional): path to the output image. Defaults to
89     """
90     # Load the image
91     img = imread(in_img)
92
93     # Find boundaries
94     red_diffusion_boundary = find_red_diffusion_boundary(img)
95     green_diffusion_boundary = find_green_diffusion_boundary(img)
96
97     # Find boundaries' coordinates
98     red_coordinates = np.where(red_diffusion_boundary == 1)
99     green_coordinates = np.where(green_diffusion_boundary == 1)
100
101     # Plot boundaries
102     img[red_coordinates[0], red_coordinates[1]] = [255, 0, 0]
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()
```