```python
from skimage.io import imread
import matplotlib.pyplot as plt
from skimage.filters import roberts
from skimage.morphology import (
    dilation,
    square,
    remove_small_objects,
    remove_small_holes,
)
import numpy as np
from pathlib import Path

ROOT = Path(__file__).parent

def find_red_diffusion_boundary(img: np.ndarray) -> np.ndarray:
    """Find the red diffusion boundary.

    Args:
        img (np.ndarray): images.

    Returns:
        np.ndarray: binary images where 1s indicate the boundary.
    """
    # Extract the red channel from the image and normalize it.
    img = img[:, :, 0] / 255.0

    # Convert the image to binary image with thresholding.
    img = img > 0.06

    #  Connected neighboring pixels by filling small holes that are less than 64 pixels.
    img = remove_small_holes(img, 64)

    # Reduce noises by eliminating connected components that are less than 20,000 pixels.
    img = remove_small_objects(img, 20000)

    # Soften edges by dilating the image with 3x3 matrix
    img = dilation(img, square(3))

    # Fill in the remaining holes
    img = remove_small_holes(img, 128)

    # Use roberts filter to find edges
    img = roberts(img)
    img = img != 0

    return img


def find_green_diffusion_boundary(img: np.ndarray) -> np.ndarray:
    """Find the green diffusion boundary.

    Args:
        img (np.ndarray): images.

    Returns:
        np.ndarray: binary images where 1s indicate the boundary.
    """
    # Extract the green channel from the image and normalize it.
    img = img[:, :, 1] / 255.0

    # Convert the image to binary image with thresholding.
    img = img > 0.06

    # Connected neighboring pixels by filling small holes that are less than 128 pixels.
    img = remove_small_holes(img, 128)

    # Reduce noises by eliminating connected components that are less than 64 pixels.
    img = remove_small_objects(img, 64)

    # Soften edges by dilating the image with 4x4 matrix
    img = dilation(img, square(4))

    # Fill in the remaining holes
    img = remove_small_holes(img, 1024)

    # Use roberts filter to find edges
    img = roberts(img)
    img = img != 0

    return img


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.

    Args:
        in_img (str, optional): path to the input image. Defaults to "./dyes.png".
        out_img (str, optional): path to the output image. Defaults to "./problem1_reuslt.png".
    """
    # Load the image
    img = imread(in_img)

    # Find boundaries
    red_diffusion_boundary = find_red_diffusion_boundary(img)
    green_diffusion_boundary = find_green_diffusion_boundary(img)

    # Find boundaries' coordinates
    red_coordinates = np.where(red_diffusion_boundary == 1)
    green_coordinates = np.where(green_diffusion_boundary == 1)

    # Plot boundaries
    img[red_coordinates[0], red_coordinates[1]] = [255, 0, 0]
    img[green_coordinates[0], green_coordinates[1]] = [0, 255, 0]

    # Save the resulting image
    plt.imsave(out_img, img)


if __name__ == "__main__":
    problem_1()
```