

```
1 from skimage.io import imread
2 from skimage.exposure import adjust_gamma
3 from skimage.morphology import dilation, square, remove_small_objects, label
4 from skimage.segmentation import flood_fill
5 from utilities import roberts, transform, unsharp_mask
6 import matplotlib.pyplot as plt
7 import numpy as np
8 from random import randint, choice
9
10 """# 1. Read "balloons.jpg" image. Find the outer edges (not the patterns inside) of the air
    balloons."""
11 # load the image and use only the green channel
12 orig_img = imread("balloons.jpg")
13 img = orig_img[:, :, 2] / 255.0
14
15 # Preprocess the image such that balloons are darker and background is uniform color.
16 img = adjust_gamma(img, gamma=2)
17 img = unsharp_mask(img)
18 img = img < 0.12
19 img = dilation(img, square(3))
20
21 # Perform edges detection with Roberts Cross
22 img = roberts(img)
23 img = img > 0
24 bin_img = remove_small_objects(img, 75)
25
26
27 """2. Count the total number of the balloons"""
28 labels_img = label(bin_img)
29 num_bln = np.max(labels_img)
30
31
32 """# 3. Plot the resulting image from step 1, and as a title of your image, write the total
    number of the balloons you found in step 2. (No hard coding please) and then save the
    resulting image as a png."""
33 fig, ax = plt.subplots(1, 2)
34 fig.suptitle(f"There are {num_bln:d} balloons in the image.", fontsize=16)
35 ax[0].set_title("Original image")
36 ax[0].imshow(orig_img)
37 ax[1].set_title("Edges image")
38 ax[1].imshow(bin_img, cmap="gray")
39 fig.savefig("3. detected_edges.png", dpi=1000)
40 plt.close()
41
42
43 """# 4. Choose a random air balloon in your binary image, change the pixels inside to white.
    Explain how you did that."""
44 # Pick a balloon based on label
45 bln = randint(1, num_bln)
46
47 # Find all pixels belong to that balloon
48 rows, cols = np.where(labels_img == bln)
49
50 # Find center the ballon
51 cen_row, cen_col = round(np.average(rows)), round(np.average(cols))
52
53 # Use flood_fill algo to fill the ballon
```

```
54 fill_img = np.copy(bin_img)
55 fill_img = flood_fill(fill_img, (cen_row, cen_col), 1)
56
57 # Save filled image
58 fig = plt.figure()
59 plt.imshow(fill_img, cmap="gray")
60 fig.suptitle(f"Balloon {bln:d} is randomly picked to be fill.")
61 fig.savefig("4. fill_a_balloons.png", dpi=1000)
62 plt.close()
63
64
65 """# 6. Move the balloon 20 pixels in any direction of 45-degree angle. Explain how you did
that."""
66 # Find all pixels belong to a ballon
67 # Fill the ballon
68 fill_img = flood_fill(labels_img, (cen_row, cen_col), bln)
69 rows, cols = np.where(fill_img == bln)
70
71 # Create a new transform image based on the original image
72 tf_img = np.copy(orig_img)
73
74 # Set transform parameters
75 tf_row = choice([-20, 20])
76 tf_col = choice([-20, 0, 20])
77 angle = 45 # in degree
78
79 # Set pixels of the balloon's original location to white
80 for row, col in zip(rows, cols):
81     tf_img[row, col, :] = 255
82
83 # Copy pixels from the balloon's original location to the new location
84 for row, col in zip(rows, cols):
85     new_row, new_col = transform(
86         (row, col), (cen_row, cen_col), (tf_row, tf_col), angle
87     )
88     if 0 <= new_row < tf_img.shape[0] and 0 <= new_col < tf_img.shape[1]:
89         tf_img[new_row, new_col, :] = orig_img[row, col, :]
90
91 # Save transform image
92 fig = plt.figure()
93 plt.imshow(tf_img)
94 fig.suptitle(
95     f"Balloon {bln:d} shifts vertically by {tf_row:d} pixels, horizontally by {tf_col:d}
pixels, and rotate by {angle:d}\N{DEGREE SIGN}",
96     fontsize=10,
97 )
98 fig.savefig("6. transform_balloons.png", dpi=1000)
99 plt.close()
100
101
102 """# 7. Rotate the air balloon 60 degrees clockwise after step 6."""
103 # Find all pixels belong to a ballon
104 # Fill the ballon
105 fill_img = flood_fill(labels_img, (cen_row, cen_col), bln)
106 rows, cols = np.where(fill_img == bln)
107
108 # Create a new transform image based on the original image
109 tf_img = np.copy(orig_img)
```

```
110 |
111 | # Set transform parameters
112 | angle = 45 + 60 # in degree
113 |
114 | # Set pixels of the balloon's original location to white
115 | for row, col in zip(rows, cols):
116 |     tf_img[row, col, :] = 255
117 |
118 | # Copy pixels from the balloon's original location to the new location
119 | for row, col in zip(rows, cols):
120 |     new_row, new_col = transform(
121 |         (row, col), (cen_row, cen_col), (tf_row, tf_col), angle
122 |     )
123 |     if 0 <= new_row < tf_img.shape[0] and 0 <= new_col < tf_img.shape[1]:
124 |         tf_img[new_row, new_col, :] = orig_img[row, col, :]
125 |
126 | # Save transform image
127 | fig = plt.figure()
128 | plt.imshow(tf_img)
129 | fig.suptitle(
130 |     f"Balloons {bln:d} shifts vertically by {tf_row:d} pixels, horizontally by {tf_col:d}
131 |     pixels, and rotate by {angle:d}\N{DEGREE SIGN}",
132 |     fontsize=10,
133 | )
134 | fig.savefig("7. rotate_balloons_again.png", dpi=1000)
135 | plt.close()
```