Homework14 Image tampering detection

胡成成 2101210578

Question

请自制COPY-MOVE图像，并尝试检测

Answer

- 代码：

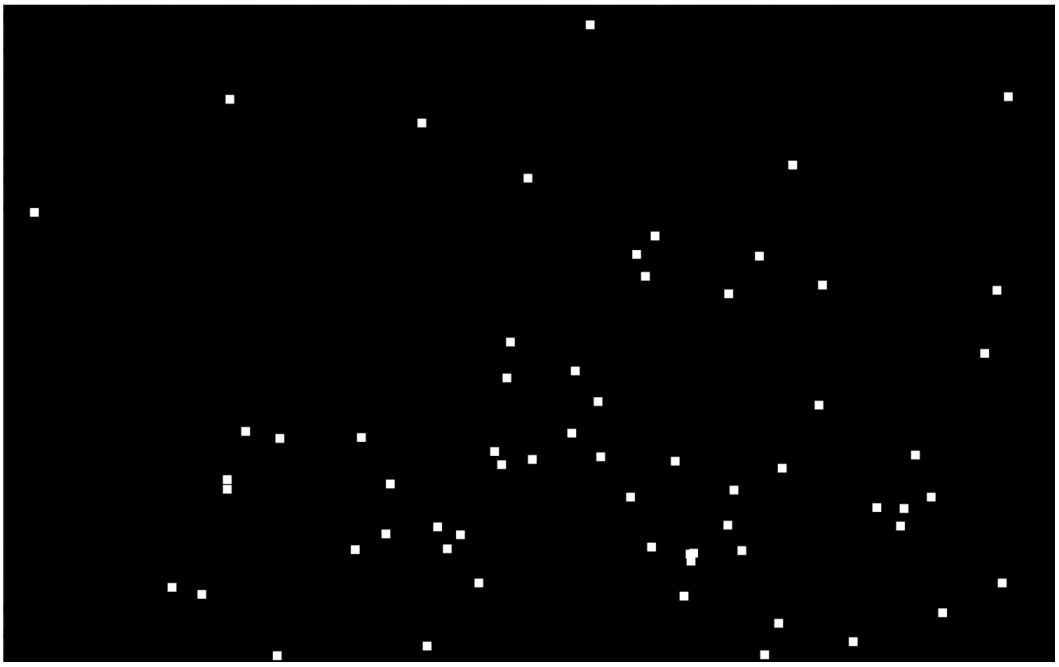
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import pywt
from matplotlib.colors import NoNorm
import cv2
import numpy as np
from scipy.spatial import distance

# initializing parameters
block_size = 10
# mean_threshold = 0.000005
# SD_threshold = 0.0000005
mean_threshold = 0.0000005
SD_threshold = 0.00000005

min_pixel_distance = 30
check_offset = 5
# reading image and provided mask ( provided mask will be used for accuracy calulation)
img = mpimg.imread('test.png')
img = np.array(img)
# converting to grayscale
img_gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
# column, row
column = img_gray.shape[1] - block_size + 1
row = img_gray.shape[0] - block_size + 1
prediction_mask = np.zeros((img_gray.shape[0], img_gray.shape[1]))
data = []
# creating array of dict objects ( with mean, SD, i, j ) by calc mean and SD of every block
print("making blocks and calculating mean with block_size = " + str(block_size))
avg_mat = np.zeros((row, column), np.int8)
block_counter = 0
for i in range(0, row):
    for j in range(0, column):
        block = img_gray[i:i + block_size, j:j + block_size]
        d = dict()
        avg_mat[i][j] = np.mean(block)
        d['M'] = np.mean(block)
        d['SD'] = np.std(block)
        d['i'] = i
        d['j'] = j
        data.append(d)
        block_counter += 1
print("Done")

# sorting according to Mean
sorted_mean = sorted(data, key=lambda element: element['M'])
# distinguishing similar blocks ( only checking neighbours in sorted array for potential similar blocks)
sim_array = []
```

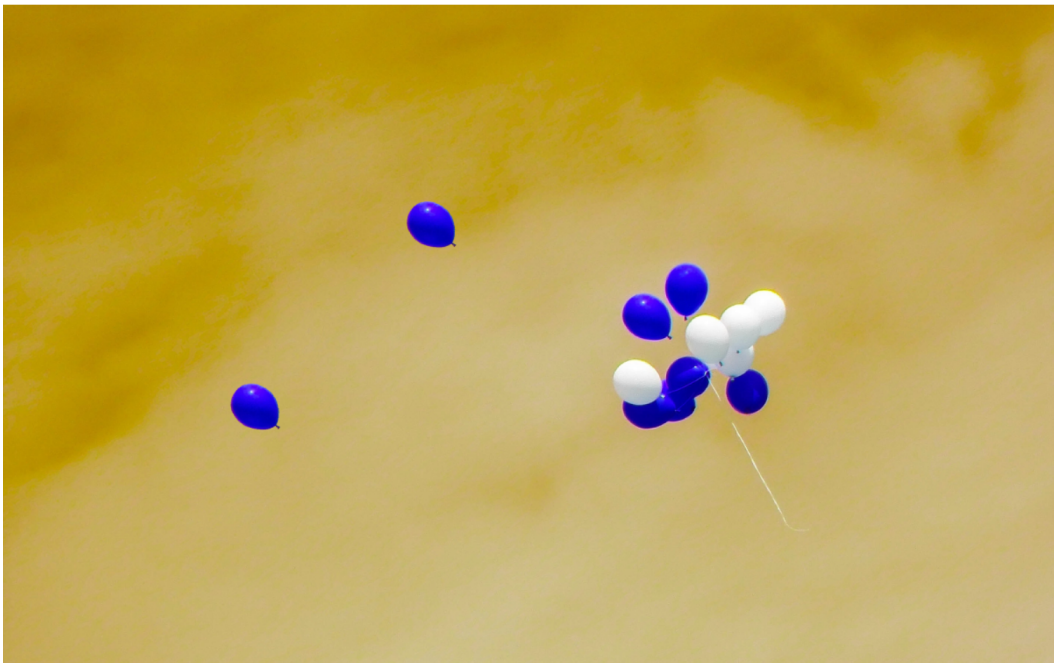
```
for i in range(len(sorted_mean)):
    for j in range(max(0, i - check_offset), min(len(sorted_mean), i + check_offset)):
        mean_similarity = abs(sorted_mean[j]['M'] - sorted_mean[i]['M'])
        SD_similarity = abs(sorted_mean[j]['SD'] - sorted_mean[i]['SD'])
        coor1 = np.array([sorted_mean[i]['i'], sorted_mean[i]['j']])
        coor2 = np.array([sorted_mean[j]['i'], sorted_mean[j]['j']])
        distance = np.linalg.norm(coor1 - coor2)
        if mean_similarity <= mean_threshold and SD_similarity <= SD_threshold and distance >= min_pixel_distance:
            sim_array.append(sorted_mean[i])
            sim_array.append(sorted_mean[j])
print(len(sim_array))
# creating prediction mask from similar blocks
for ele in sim_array:
    i = ele['i']
    j = ele['j']
    prediction_mask[i:i + block_size, j:j + block_size] = 255

# plt.figure(1)
cv2.imshow("copied", prediction_mask)
cv2.imshow("img", img)
# plt.imshow(prediction_mask,cmap="gray")

# plt.show()
cv2.waitKey(0)
cv2.DestroyAllWindows()
```

- 测试原始图片

- PhotoShop印章工具制作图片

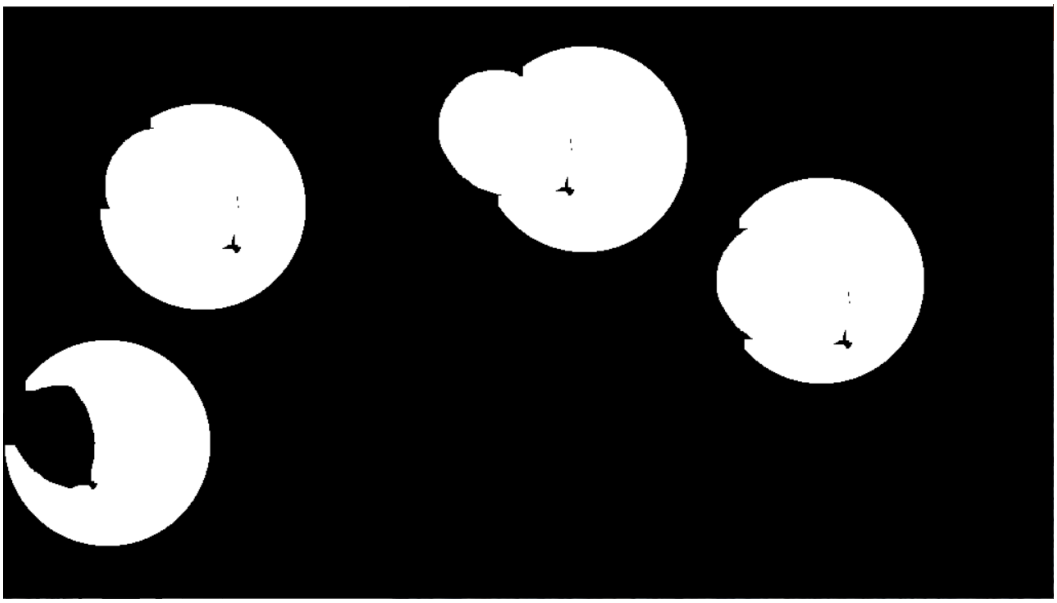
- 运行结果图

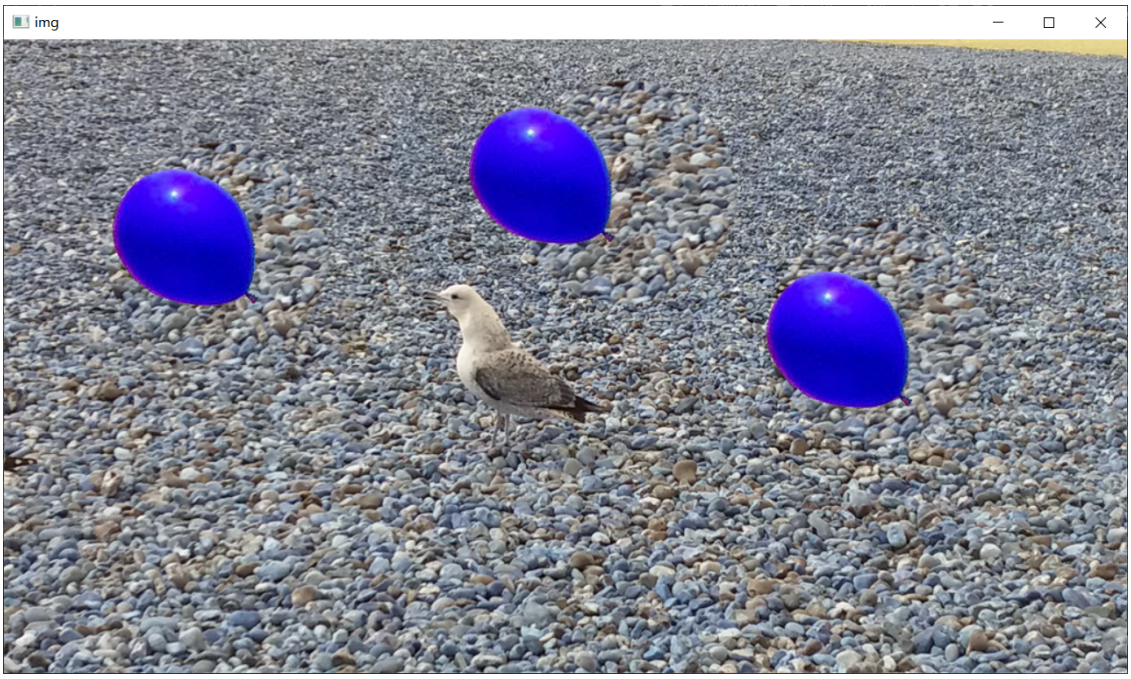




- 发现识别的效果并是很好，可能是吧背景色彩太过一致，于是加了个混杂一点的背景，把三个一样的气球放上去：

- 识别的效果如下图所示：




img