实验四 图像特征提取-1

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名：陈志鸿** | **学号：2020281024** | **日期：2021-11-23** | **截止日期：Next Class** |

**一、实验目的**

1. 掌握Opencv图像颜色空间变换操作
2. 掌握Opencv提取图像的矩特征

**二、实验报告要求**

1. 实验报告中，**按题目顺序给出实验内容对应的代码及相应结果**，**体现实验步骤和方法**。要求报告整洁，程序清晰，代码简捷，有必要的注释
2. **报告中的图片**要求清晰可见。 如果是屏幕截图，**请不要截取有效结果之外的无关区域**。
3. **报告中的代码**，一般情况下紧随内容。如果代码太长，建议按题目序号，附在报告末尾。

**三、实验内容及要求**

**一、概念题**

1. RGB图像中的R,G,B分别代表什么含义？

红、绿、蓝通道

2. 图像颜色空间HSV中的H,S,V分别代表什么？

色调、饱和度、明度

3. 简述如何产生归一化一阶差分链码来描述一个边界？为什么要这么做？

先获得边界点，再通过对点的四领域或者八领域进行编号得到归一化链码，计算两个链码之间的距离得到归一化差分链码。因为差分链码具有具有平移不变性和旋转不变性，利于对图像进行描述。

4. 从图像中提取的特征具有平移、灰度、尺度、旋转不变性，是什么意思？

这是对一张图片的特征的描述，是固定性质，图片特征不会随着尺度、颜色等变化而改变

5. 如何求得一幅图像的质心？

在x方向和y方向上分别独立地找出质心，由零阶原点矩和一阶原点矩结合求出，x轴坐标为M10 / M00，y轴坐标为M01 / M00

二、**编程题L1**

2.1 **颜色空间转换**

编写函数P4\_rgb2gray() 实现彩色图到灰度图的转换，转换公式如下

Gray(i,j) = 0.299 \* R(i,j) + 0.587 \* G(i,j) + 0.144 \* B(i,j);

其中R,G,B 分别为彩色图像的RGB通道。

1. **import** numpy as np
2. **import** cv2 as cv
4. **def** P4\_rgb2gray(img) :
5. img = 0.229 \* img[ : , : , 2] + 0.578 \* img[ : , : , 1] + 0.144 \* img[ : , : , 0]
6. **return** img
8. img = cv.imread('images/shenzhen.png', cv.IMREAD\_COLOR)
9. img\_gray = P4\_rgb2gray(img).astype(np.uint8)
11. cv.imshow('img\_gray', img\_gray)
12. cv.waitKey(0)



2.2 **利用颜色进行目标识别**

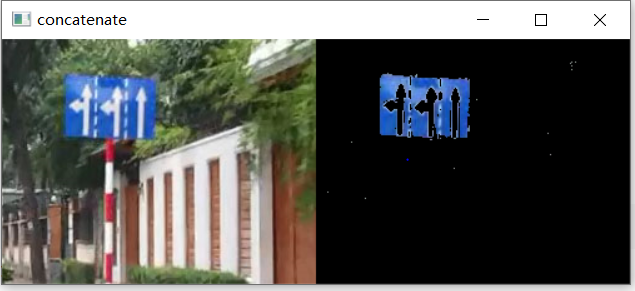
目的：检测图像blue\_sign.png 中的蓝色标志牌。

* 方法1. 如果使用灰度图中，是否可以检测该蓝色标志牌？为什么？

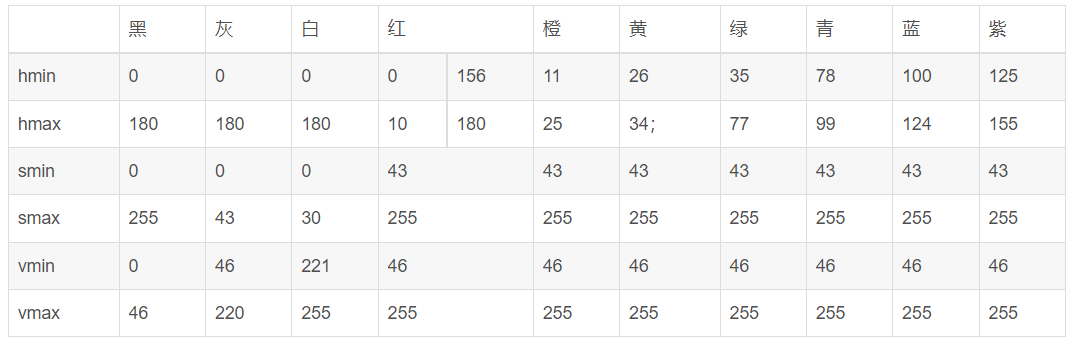
不能检测，因为灰度图只有一个通道，蓝色无法得到表现。

* 方法2. 如果使用RGB彩色图，是否可以检测该蓝色标志牌？怎么检测？一个可能的办法是，可以设定(R,G,B)的范围，这个范围内的像素呈现蓝色，然后把不在这个范围内的像素置成黑色。如果不能检测，为什么？

1. **import** cv2 as cv
2. **import** numpy as np
4. img = cv.imread('images/blue\_sign.png', cv.IMREAD\_COLOR)
5. rgb = cv.cvtColor(img, cv.COLOR\_BGR2RGB)
7. blue\_low = np.array([0, 0, 150])
8. blue\_up = np.array([150, 240, 255])
10. mask = cv.inRange(rgb, blue\_low, blue\_up)
11. img2 = cv.bitwise\_and(img, img, mask = mask)
13. concatenate = np.concatenate((img, img2), axis = 1)
14. cv.imshow('concatenate', concatenate)
15. cv.waitKey(0)



* 方法3. 如果使用HSV彩色空间，是否可以检测该蓝色标志牌？和方法2同样的思路，找到一个区间，去过滤像素。下图可以用来确定不同颜色对应H，S，V分量的区间。在HSV颜色空间里，是否可以检测？



**提示：**用数值区间过滤像素，请查阅函数cv2.inRange()

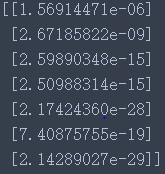
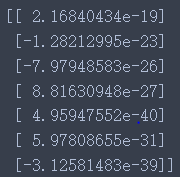
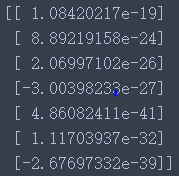
1. **import** cv2 as cv
2. **import** numpy as np
3. **import** matplotlib.pyplot as plt
4. img = cv.imread('images/blue\_sign.png')[ : , : , : : -1]
5. hsv = cv.cvtColor(img, cv.COLOR\_RGB2HSV)
7. low\_blue = np.array([100, 43, 46])
8. up\_blue = np.array([124, 255, 240])
10. mask = cv.inRange(hsv, low\_blue, up\_blue)
11. img = cv.cvtColor(img, cv.COLOR\_RGB2BGR)
12. img2 = cv.bitwise\_and(img, img, mask = mask)
14. concatenate = np.concatenate((img, img2), axis = 1)
15. cv.imshow('concatenate', concatenate)
16. cv.waitKey(0)



2.3 **验证Hu矩的不变性**

计算star.png所示目标的Hu矩，并验证Hu矩具有平移，旋转，尺度不变性。

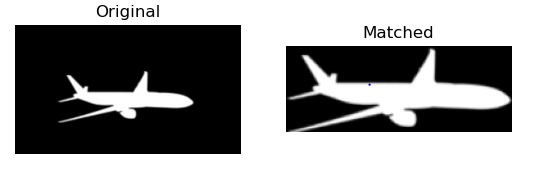
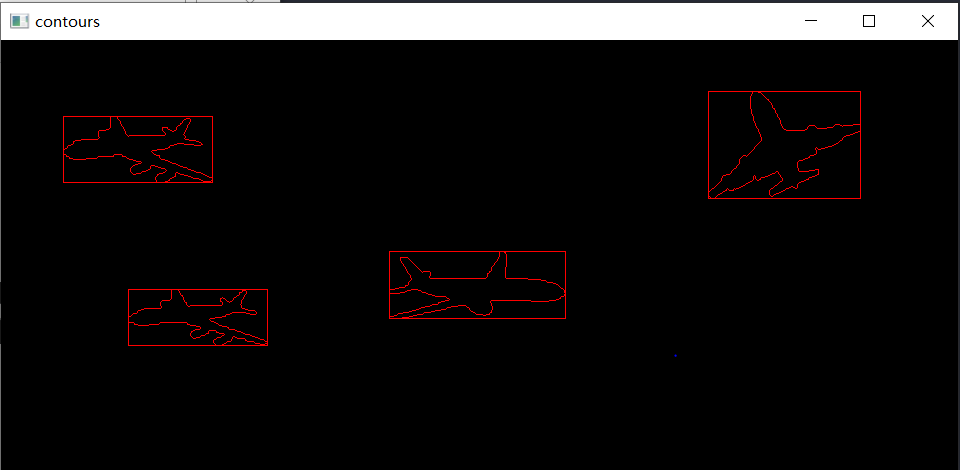
1. **import** numpy as np
2. **import** cv2 as cv
3. star\_img = cv.imread('images/star.png', cv.IMREAD\_GRAYSCALE)
4. moments = cv.moments(star\_img)
5. HuMoments = cv.HuMoments(moments)
7. #平移
8. M = np.float32([[1, 0, 50], [0, 1, 30]])
9. rows, cols = star\_img.shape[0], star\_img.shape[1]
10. star\_img1 = cv.warpAffine(star\_img, M, (cols, rows))
11. moments = cv.moments(star\_img1)
12. HuMoments1 = cv.HuMoments(moments)
13. delta1 =  HuMoments1 - HuMoments
14. **print**(delta1)
16. #旋转
17. star\_img2 = np.rot90(star\_img)
18. moments = cv.moments(star\_img2)
19. HuMoments2 = cv.HuMoments(moments)
20. delta2 = HuMoments2 - HuMoments
21. **print**(delta2)
23. #尺度
24. star\_img3 = cv.resize(star\_img, (300, 300))
25. moments = cv.moments(star\_img3)
26. HuMoments3 = cv.HuMoments(moments)
27. delta3 = HuMoments3 - HuMoments
28. **print**(delta3)

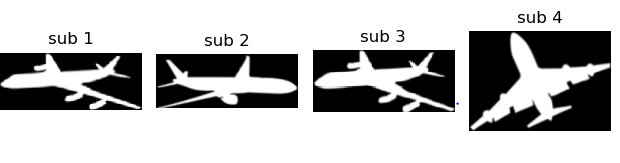


2.4 **利用图像目标的Hu矩进行形状匹配**

目的：在图像planes.png中定位出one\_plane.png 所示目标的位置，并画出其bounding box。大致步骤如下

1. 查找planes.png中所有目标的轮廓
2. 对每一个轮廓求得其最小的外接矩形 (bounding box)，得到各个飞机的小图
3. 对one\_plane.png 和步骤2中的每个小图计算Hu矩，并计算Hu矩的距离。Hu矩最相近的小图应该是目标所在。
4. 在planes.png 中画出矩形定位框。使用函数cv2.rectangle(img, pt1, pt2, 255, 2), 其中pt1为bounding box 的左上角坐标，pt2 为右下角坐标。
5. **import** cv2 as cv
6. **import** numpy as np
7. **import** matplotlib.pyplot as plt
8. %matplotlib
9. img\_planes = cv.imread('images/planes.png', 0)
11. img = np.zeros((img\_planes.shape[0], img\_planes.shape[1], 3))
12. contours, hierarphy = cv.findContours(img\_planes, cv.RETR\_EXTERNAL, cv.CHAIN\_APPROX\_NONE)#contours得到宽和高
13. cv.drawContours(img, contours,  -1, (0, 0, 255), 1)
15. contours\_tmp = []
16. **for** k **in** range(len(contours)) :
17. X = contours[k].squeeze()
18. min\_x, min\_y, max\_x, max\_y = np.min(X, 0)[0], np.min(X, 0)[1], np.max(X, 0)[0], np.max(X, 0)[1]
19. contours\_tmp.append([[min\_x, min\_y], [min\_x, max\_y], [max\_x, max\_y], [max\_x, min\_y]])
20. #cv2.line画矩形
21. # for i in range(len(contours\_tmp)) :
22. #     cv.line(img, (contours\_tmp[i][3][0], contours\_tmp[i][3][1]), (contours\_tmp[i][0][0], contours\_tmp[i][0][1]), (0, 0, 255), 1)
23. #     for j in range(len(contours\_tmp[i]) - 1) :
24. #         cv.line(img, (contours\_tmp[i][j][0], contours\_tmp[i][j][1]), (contours\_tmp[i][j + 1][0], contours\_tmp[i][j + 1][1]), (0, 0, 255), 1)
25. #         cv.imshow('img', img)
26. #         cv.waitKey(1)
27. # cv.waitKey(0)
28. **for** i **in** range(len(contours\_tmp)) :
29. cv.rectangle(img, (contours\_tmp[i][0][0], contours\_tmp[i][0][1]), (contours\_tmp[i][2][0], contours\_tmp[i][2][1]), (0, 0, 255))
31. img\_plane = cv.imread('images/one\_plane.png', 0)
32. sub\_images = []
33. deltas = []
34. #单架飞机的Hu矩
35. moments\_plane = cv.moments(img\_plane)
36. Hu\_plane = cv.HuMoments(moments\_plane)
37. Hu\_planes = []
39. **for** k **in** range(len(contours\_tmp)) :
40. sub\_image = img\_planes[contours\_tmp[k][0][1] : contours\_tmp[k][2][1], contours\_tmp[k][0][0] : contours\_tmp[k][2][0]].astype(np.float32)
41. sub\_images.append(sub\_image)
43. moments = cv.moments(sub\_image)
44. Humoments = cv.HuMoments(moments)
45. Hu\_planes.append(Humoments)
46. #定义Hu矩的距离为欧几里得距离
47. delta = np.square(Humoments - Hu\_plane)
48. deltas.append(np.sum(delta))
50. plt.subplot(1, 4, k + 1)
51. plt.title(f'sub {k + 1}')
52. plt.axis('off')
53. plt.tight\_layout()
54. plt.imshow(sub\_image, 'gray')
56. plt.figure()
57. the\_nearest = deltas.index(min(deltas))
58. plt.subplot(1, 2, 1), plt.imshow(img\_plane, 'gray'), plt.title('Original'),plt.axis('off')
59. plt.subplot(1, 2, 2), plt.imshow(sub\_images[the\_nearest], 'gray'), plt.title('Matched'),plt.axis('off')
61. cv.imshow('contours', img)
62. cv.waitKey(0)





三、**编程题 L2**

3.1 **编写函数实现Freeman边界编码**

利用cv.findContours 产生star.png 的边界，编写函数P4\_contour\_code(pointList) 生成8领域归一化一阶差分链码, 函数的输入是边界点序列，给出图中所示目标的边界编码。

1. **import** cv2 as cv
2. **def** P4\_contour\_code(pointList) :
3. Normalized\_chain\_code = []
4. Differential\_code = []
5. direction = [[1, 0], [1, 1], [0, 1], [-1, 1], [-1, 0], [-1, -1], [0, -1], [1, -1]]
7. **for** k **in** range(1, len(contours\_tmp)) :
8. **for** t **in** range(len(direction)) :
9. **if** (contours\_tmp[k][0] == (contours\_tmp[k - 1][0] + direction[t][0])) **and** (contours\_tmp[k][1] == (contours\_tmp[k - 1][1] + direction[t][1])) :
10. Normalized\_chain\_code.append(t)
11. Normalized\_chain\_code.append(Normalized\_chain\_code[0])
13. **for** i **in** range(1, len(Normalized\_chain\_code)) :
14. **if** Normalized\_chain\_code[i] < Normalized\_chain\_code[i - 1] :
15. Differential\_code.append(Normalized\_chain\_code[i] + 8 - Normalized\_chain\_code[i - 1])
16. **else** :
17. Differential\_code.append(Normalized\_chain\_code[i] - Normalized\_chain\_code[i - 1])
18. **return** Differential\_code, Normalized\_chain\_code
20. star = cv.imread('images/star.png', cv.IMREAD\_GRAYSCALE)
21. contours, hierarphy = cv.findContours(star, cv.RETR\_EXTERNAL, cv.CHAIN\_APPROX\_NONE)
22. pointList = contours[0].squeeze()
24. Chain\_code, Differential\_code = P4\_contour\_code(pointList)
25. **print**('归一化链码: \n',"".join(map(str, Chain\_code)))
26. **print**('归一化差分码: \n',"".join(map(str, Differential\_code)))

