实验三 图像的二值化和形态学处理

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名：陈志鸿** | **学号：2020281024** | **日期：2021-11-09** | **截止日期：Next Class** |

**一、实验目的**

1. 掌握Opencv图像二值化的基本方法和操作
2. 掌握Opencv图像形态学操作的基本原理及方法

**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. 图像微分 增强 了边缘和其它突变的信息。（填“增强”或“削弱”）
2. 图像微分算子 能 用在边缘检测中。（填“能”或“不能”）
3. 形态学处理中最基本的运算是腐蚀与膨胀。其中， 腐蚀 通常在去除小颗粒以及消除目标物之间的粘连是非常有效的。 膨胀 通常用以填补目标物中存在的某些空洞。
4. 开运算是使用同一个结构元素对图像先 腐蚀 再进行膨胀的运算。
5. 闭运算是使用同一个结构元素对图像先 膨胀 再进行腐蚀的运算。

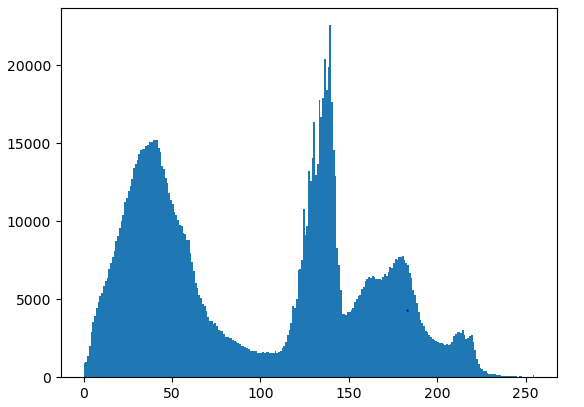
**6. 简答：**Otsu法或最大类间方差法的基本原理是什么？

以某个灰度值为阈值，将图片二值化，然后计算两组灰度值之间的方差以及组内的方差，然后以此为标准对阈值进行迭代，找到一个值能够使组间方差/组内方差的值最大，这个值就是最优解

二、编程题Level 1

2.1 针对house.jpg 图像，考察其灰度直方图，如果采用双峰法来二值化图像，阈值多少较为合适？并给出该参数下THRESH\_BINARY\_INV， THRESH\_TRUNC，THRESH\_TOZERO 的二值化效果图。

1. **import** cv2 as cv
2. **import** numpy as np
3. **import** matplotlib.pyplot as plt
4. %matplotlib
6. img = cv.imread('images/house.jpg', cv.IMREAD\_GRAYSCALE)
7. # img = cv.resize(img, (700, 500))
8. plt.hist(img.ravel(), bins = 255)
9. plt.show()
11. res, img1 = cv.threshold(img, 100, 255, cv.THRESH\_BINARY\_INV)
12. res, img2 = cv.threshold(img, 100, 255, cv.THRESH\_TRUNC)
13. res, img3 = cv.threshold(img, 100, 255, cv.THRESH\_TOZERO)
15. plt.figure()
16. images = [img, img1, img2, img3]
17. title = ['house', 'THRESH\_BINARY\_INV', 'THRESH\_TRUNC', 'THRESH\_TOZERO']
18. **for** i **in** range(4) :
19. plt.subplot(2, 2, i + 1)
20. plt.imshow(images[i], 'gray')
21. plt.title(title[i])
22. plt.tight\_layout()
23. plt.xticks([]), plt.yticks([])
24. plt.show()



2.2 形态学顶帽是原图像与开运算（先腐蚀后膨胀）之差，得到的是原图像经过开运算之后去掉的像素，目的是为了在暗的背景图像中增强亮的目标物体。黑帽是闭运算（先膨胀后腐蚀）与原图像之差，得到的是原图像经过开运算之后增加的像素，目的是为了在亮的背景图像中增强暗的目标物体。以wall.png 为列，给出形态学顶帽，黑帽的效果。

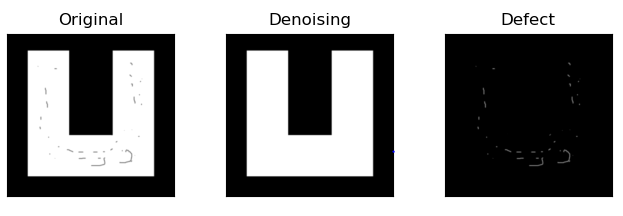
**提示：**使用cv2.morphologyEx, 函数和cv.MORPH\_TOPHAT, cv2.MORPH\_BLACKHAT 参数。

1. **import** cv2 as cv
2. **import** numpy as np
3. **import** matplotlib.pyplot as plt
4. %matplotlib
5. img = cv.imread('images/wall.png', cv.IMREAD\_GRAYSCALE)
6. kernel = np.ones((10, 10)).astype(np.int32)
7. Tophat = cv.morphologyEx(img, cv.MORPH\_TOPHAT, kernel)
8. Blackhat = cv.morphologyEx(img, cv.MORPH\_BLACKHAT, kernel)
9. title = ['Wall', 'TopHat', 'BlackHat']
10. images = [img, Tophat, Blackhat]
11. **for** i **in** range(3) :
12. plt.subplot(1, 3, i + 1)
13. plt.imshow(images[i], 'gray')
14. plt.title(title[i])
15. plt.tight\_layout()
16. plt.xticks([]), plt.yticks([])
17. plt.show()



2.3 针对morph\_test.png, 图中的曲线可视为产品上的缺陷。 运用形态学算子，做缺陷检测。即，在一张新的黑白图中画出缺陷。

1. **import** cv2 as cv
2. **import** matplotlib.pyplot as plt
3. **import** numpy as np
4. %matplotlib
5. img = cv.imread('images/morph\_test.png', cv.IMREAD\_GRAYSCALE)
6. kernel = np.ones((7, 7)).astype(np.int32)
7. img1 = cv.morphologyEx(img, cv.MORPH\_CLOSE, kernel)
8. img2 = cv.bitwise\_xor(img, img1)
9. title = ['Original', 'Denoising', 'Defect']
10. images = [img, img1, img2]
11. **for** i **in** range(3) :
12. plt.subplot(1, 3, i + 1)
13. plt.imshow(images[i], 'gray')
14. plt.title(title[i])
15. plt.tight\_layout()
16. plt.xticks([]), plt.yticks([])
17. plt.show()



三、编程题 Level 2

3.1编写函数img2, thd = P3\_1(img) 实现Otsu法或最大类间方差法，把图像分割成前景和背景，画出类间方差同分割阈值的变化曲线，并给出house.jpg图片测试结果。函数签名中，img2为二值分割图像，背景为0，前景为255，thd 为分割的阈值。把你的结果同opencv 函数

ret, img2 = cv2.threshold(img,0,255,cv2.THRESH\_BINARY + cv2.THRESH\_OTSU) 相比较。

1. **import** copy
2. **import** cv2 as cv
3. **import** matplotlib.pyplot as plt
4. **import** numpy as np
5. %matplotlib
7. house = cv.imread('images/house.jpg', cv.IMREAD\_GRAYSCALE)
8. house = cv.resize(house, (700, 500))
9. **def** P3\_1(img) :
10. cimg = copy.deepcopy(img)
11. Res = np.ones(255)
12. std = np.zeros(255)
13. **for** thd **in** range(0, 255) :
14. arr1 = cimg[np.where(img > thd)]
15. arr2 = cimg[np.where(img <= thd)]
16. w1 = len(arr1)
17. w2 = len(arr2)
18. m1 = np.sum(arr1) / w1
19. m2 = np.sum(arr2) / w2
20. m = (m1 \* w1 + m2 \* w2) / (w1 + w2)
21. #组内方差和
22. std\_ingroud = np.sum(np.square(arr1 - m1)) + np.sum(np.square(arr2 - m2))
23. #组间方差和
24. std\_outgroud = (m1 - m) \* (m1 - m) \* w1 + (m2 - m) \* ( m2 - m) \* w2
25. #同时考虑组内和组间方差，当组内方差越小、组间方差越大时效果越好。
26. res = std\_outgroud / std\_ingroud
27. std[thd] = std\_outgroud
28. Res[thd] = res
29. plt.plot(list(range(255)), Res, 'r-')
30. plt.show()
31. thd = np.argmax(Res[ : ])
32. cimg[np.where(cimg >= thd)] = 255
33. cimg[np.where(cimg < thd)] = 0
34. **return** cimg, thd
36. img2, thd = P3\_1(house)
37. ret, img3 = cv.threshold(house, 0, 255, cv.THRESH\_BINARY + cv.THRESH\_OTSU)
39. images = [house, img2, img3]
40. title = ['original', f'my threshold {thd}', f'cv threshold {int(ret)}']
41. **for** i **in** range(3) :
42. plt.subplot(1, 3, i + 1)
43. plt.imshow(images[i], 'gray')
44. plt.title(title[i])
45. plt.xticks([]), plt.yticks([])
46. plt.tight\_layout()
47. plt.show()
48. **print**(f'thd : {thd}, ret : {ret}'





四、综合题

4.1 针对PCB.png 做blob分析。 请给出解决方案，提取焊点的大概位置，并统计焊点的个数。焊点在图中呈现白色，如下图所示。



**提示和建议：**

1. 以彩色的方式读取图片 （cv2.IMREAD\_COLOR），选取第三个通道作后续的处理。
2. 选取合适的结构做开或者闭运算，以及二值化操作。
3. **import** cv2 as cv
4. **import** matplotlib.pyplot as plt
5. **import** numpy as np
6. %matplotlib
8. img = cv.imread('images/PCB.png', cv.IMREAD\_COLOR)
9. img = cv.cvtColor(img, cv.COLOR\_BGR2RGB)
10. r, g, b = cv.split(img)
12. kernel1 = np.zeros((16, 16)).astype(np.int32)
13. kernel2 = np.zeros((6, 6)).astype(np.int32)
14. #圆心
15. center\_radius1 = int(16 / 2)
16. center\_radius2 = int(6 / 2)
17. #做圆形卷积核
18. cv.circle(kernel1, (center\_radius1, center\_radius1), center\_radius1, (255, 255, 255), -1, cv.LINE\_AA)
19. cv.circle(kernel2, (center\_radius2, center\_radius2), center\_radius2, (255, 255, 255), -1, cv.LINE\_AA)
21. kernel1 = kernel1.astype(np.uint8)
22. kernel2 = kernel2.astype(np.uint8)
24. \_, img\_binary = cv.threshold(b, 180, 255, cv.THRESH\_BINARY)
25. img\_open = cv.morphologyEx(img\_binary, cv.MORPH\_OPEN, kernel2)
26. img\_dilate = cv.dilate(img\_close, kernel1, 10)
28. #焊点个数
29. Solder\_joint = 0
30. #直线检测，以行为迭代对象，如果如果遇到白点则记录到直线中
31. #因为焊点之间是分开的，则直线中不为零的某一段就代表一个焊点
32. Detection\_line = np.zeros(img\_dilate.shape[1])
33. **for** row **in** range(img\_dilate.shape[0]) :
34. **if** row == img\_dilate.shape[0] / 2 **or** row == img\_dilate.shape[0] - 1:#总统计焊点两次
35. flag = 0
36. **for** i **in** range(len(Detection\_line)) :
37. **if** Detection\_line[i] == 0 **and** flag == 1 :#如果当前点是黑点且是由白到黑点的跨越，则代表着一个焊点
38. Solder\_joint += 1
39. flag = 0
40. **if** Detection\_line[i] != 0 :#由白点进入到黑点，记录flag为1
41. flag = 1
42. Detection\_line[ : ] = 0#统计完一次黑点对直线清零
43. Detection\_line = Detection\_line + img\_dilate[row]
45. **print**(f'焊点个数:{Solder\_joint}')
46. cv.imshow('img\_dilate', img\_dilate)
47. cv.waitKey(0)



