作業3:Morphology Boundary extraction and Region Filling

班級: 碩專班

學生:5108056016張仲威

教授: 吳俊霖 教授

目錄

[一. 主要演算法 2](#_Toc60878117)

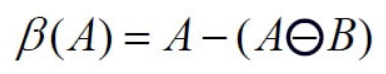
[二. 程式片斷 2](#_Toc60878118)

[三. 測試資料 7](#_Toc60878119)

[四. 結果呈現 7](#_Toc60878120)

[五. 結論 8](#_Toc60878121)

# 主要演算法



第一個邊界抽取 Boundary Extraction概念比較簡單, 用 **A 跟 B 做 Erosion** 會得到一個比較小的A，再用**原來的A 減掉比較小的A**，那麼多出來的地方就剛好是邊界了



第二個Region Filling比較複雜, Xk 為你要填充區域裡面的Element，B為要拿來做**Dilation** 的Structures Element，配合後面**跟A 的補集做 AND 的條件**，然後就會從 k = 1、2、3、... 開始做，做到被 A 包住的整個範圍都做完以後結束

# 程式片斷

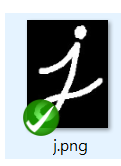
1. Boundary Extraction:
2. **def** doErosion(binary\_img\_matrix = None, structuring\_element = default\_structure):
3. binary\_img\_matrix = np.asarray(binary\_img\_matrix)
4. structuring\_element = np.asarray(structuring\_element)
5. ste\_shp = structuring\_element.shape
6. eroded\_img = np.zeros((binary\_img\_matrix.shape[0], binary\_img\_matrix.shape[1]))
7. ste\_origin = (int(np.ceil((structuring\_element.shape[0] - 1) / 2.0)), int(np.ceil((structuring\_element.shape[1] - 1) / 2.0)))
8. **for** i **in** range(len(binary\_img\_matrix)):
9. **for** j **in** range(len(binary\_img\_matrix[0])):
10. overlap = binary\_img\_matrix[idx\_check(i - ste\_origin[0]):i + (ste\_shp[0] - ste\_origin[0]),
11. idx\_check(j - ste\_origin[1]):j + (ste\_shp[1] - ste\_origin[1])]
12. shp = overlap.shape
13. ste\_first\_row\_idx = int(np.fabs(i - ste\_origin[0])) **if** i - ste\_origin[0] < 0 **else** 0
14. ste\_first\_col\_idx = int(np.fabs(j - ste\_origin[1])) **if** j - ste\_origin[1] < 0 **else** 0
16. ste\_last\_row\_idx = ste\_shp[0] - 1 - (i + (ste\_shp[0] - ste\_origin[0]) - binary\_img\_matrix.shape[0]) **if** i + (ste\_shp[0] - ste\_origin[0]) > binary\_img\_matrix.shape[0] **else** ste\_shp[0]-1
17. ste\_last\_col\_idx = ste\_shp[1] - 1 - (j + (ste\_shp[1] - ste\_origin[1]) - binary\_img\_matrix.shape[1]) **if** j + (ste\_shp[1] - ste\_origin[1]) > binary\_img\_matrix.shape[1] **else** ste\_shp[1]-1
19. **if** shp[0] != 0 **and** shp[1] != 0 **and** np.array\_equal(np.logical\_and(overlap, structuring\_element[ste\_first\_row\_idx:ste\_last\_row\_idx+1,
20. ste\_first\_col\_idx:ste\_last\_col\_idx+1]),structuring\_element[ste\_first\_row\_idx:ste\_last\_row\_idx+1,
21. ste\_first\_col\_idx:ste\_last\_col\_idx+1]):
22. eroded\_img[i, j] = 1
23. **return** eroded\_img
24. img = cv2.imread('./img/j.png',0) # 原圖
25. erosion = doErosion(binary\_img\_matrix=img) # 原圖和Structures Element erosion結果
27. img\_float=img\_as\_float(img)
28. boundary = cv2.subtract(img\_float,erosion) # 原圖-erosion結果

2. Region Filling:

1. **def** doDilation(binary\_img\_matrix = None, structuring\_element = default\_structure):
2. binary\_img\_matrix = np.asarray(binary\_img\_matrix)
3. structuring\_element = np.asarray(structuring\_element)
4. ste\_shp = structuring\_element.shape
5. dilated\_img = np.zeros((binary\_img\_matrix.shape[0], binary\_img\_matrix.shape[1]))
6. ste\_origin = ((structuring\_element.shape[0]-1)/2, (structuring\_element.shape[1]-1)/2)
7. **for** i **in** range(len(binary\_img\_matrix)):
8. **for** j **in** range(len(binary\_img\_matrix[0])):
9. a=int(idx\_check(i - ste\_origin[0]))
10. b=int(i + (ste\_shp[0] - ste\_origin[0]))
11. c=int(idx\_check(j - ste\_origin[1]))
12. d=int(j + (ste\_shp[1] - ste\_origin[1]))
13. overlap = binary\_img\_matrix[a:b, c:d]
14. shp = overlap.shape
16. ste\_first\_row\_idx = int(np.fabs(i - ste\_origin[0])) **if** i - ste\_origin[0] < 0 **else** 0
17. ste\_first\_col\_idx = int(np.fabs(j - ste\_origin[1])) **if** j - ste\_origin[1] < 0 **else** 0
19. ste\_last\_row\_idx = ste\_shp[0] - 1 - (i + (ste\_shp[0] - ste\_origin[0]) - binary\_img\_matrix.shape[0]) **if** i + (ste\_shp[0] - ste\_origin[0]) > binary\_img\_matrix.shape[0] **else** ste\_shp[0]-1
20. ste\_last\_col\_idx = ste\_shp[1] - 1 - (j + (ste\_shp[1] - ste\_origin[1]) - binary\_img\_matrix.shape[1]) **if** j + (ste\_shp[1] - ste\_origin[1]) > binary\_img\_matrix.shape[1] **else** ste\_shp[1]-1
21. e=int(ste\_first\_row\_idx)
22. f=int(ste\_last\_row\_idx+1)
23. g=int(ste\_first\_col\_idx)
24. h=int(ste\_last\_col\_idx+1)
25. **if** shp[0] != 0 **and** shp[1] != 0 **and** np.logical\_and(structuring\_element[e:f, g:h], overlap).any():
26. dilated\_img[i, j] = 1
27. **return** dilated\_img
28. img = cv.imread("./img/regionfilling.jpg")
30. # 二值化
31. imgray = cv.cvtColor(img, cv.COLOR\_BGR2GRAY)
32. imgray[imgray < 100] = 0
33. imgray[imgray >= 100] = 255
35. # 原圖取補
36. mask = 255 - imgray
38. # 定義Marker圖
39. marker = np.zeros\_like(imgray)
40. marker[0, :] = 255
41. marker[-1, :] = 255
42. marker[:, 0] = 255
43. marker[:, -1] = 255
44. marker\_0 = marker.copy()
46. # 形態學重建
47. SE = cv.getStructuringElement(shape=cv.MORPH\_CROSS, ksize=(3, 3))
48. # 無窮迴圈,做到XK不在改變
49. **while** True:
50. marker\_pre = marker
51. dilation = doDilation(marker, structuring\_element=SE)
52. marker = np.min((dilation, mask), axis=0)
53. **if** (marker\_pre == marker).all():
54. **break**
55. dst = 255 - marker
56. filling = dst - imgray
58. imgray\_float=img\_as\_float(imgray)
59. result = cv.subtract(dst,imgray\_float)
60. # 畫圖
61. plt.figure(figsize=(12, 6))  # width \* height
62. plt.subplot(2, 3, 1), plt.imshow(imgray, cmap='gray'), plt.title('Original image'), plt.axis("off")
63. plt.subplot(2, 3, 2), plt.imshow(mask, cmap='gray'), plt.title('Original image ^'), plt.axis("off")
64. plt.subplot(2, 3, 3), plt.imshow(marker\_0, cmap='gray'), plt.title('Marker 0'), plt.axis("off")
65. plt.subplot(2, 3, 4), plt.imshow(marker, cmap='gray'), plt.title('Marker'), plt.axis("off")
66. plt.subplot(2, 3, 5), plt.imshow(dst, cmap='gray'), plt.title('dst'), plt.axis("off")
67. plt.subplot(2, 3, 6), plt.imshow(result, cmap='gray'), plt.title('Region Filling result'), plt.axis("off")
68. plt.show()

# 測試資料

1.Boundary Extraction:



2. Region Filling:

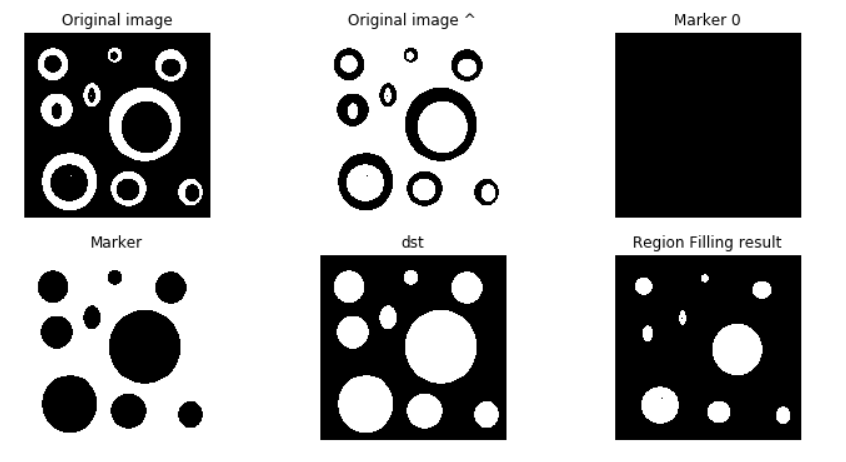


# 結果呈現

1.Boundary Extraction:



1. Region Filling



# 結論

此作業讓我了解簡單的影像形態學操作Dilation或Erosion,能應用於邊界抽取和區域填充,覺得這樣的作法也是可以讓結果做得很好