國立中興大學

一一一學年度第一學期

數位影像處理

第四次平時作業

班級：資工碩士一年級
學號：7110056210
學生：丁吾心
授課教師：吳俊霖 教授

目錄

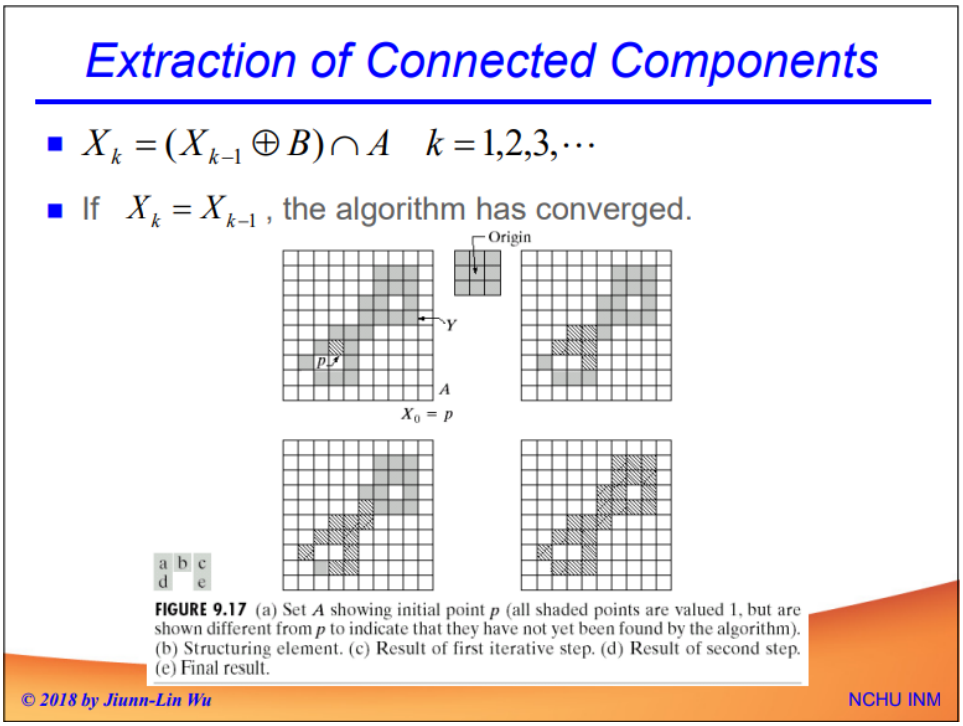
連通域 Connect component 2

1.1 題目說明 2
1.2 演算法流程 3
1.3 詳細流程與說明 4
1.3.1 二值化 4
1.3.2 遞迴主函式 4
1.3.3 遞迴式 5
1.4 結果 5
1.5 程式碼 6
1.6 心得分享 8

連通域 Connect component

1.1 題目說明

一般是指影像中具有相同像素值且位置相鄰(四鄰接、八鄰接..)的像素點組成的影像區域，本次作業要算出一張影像中有幾個連通域，以及各個連通域中分別有幾個像素。

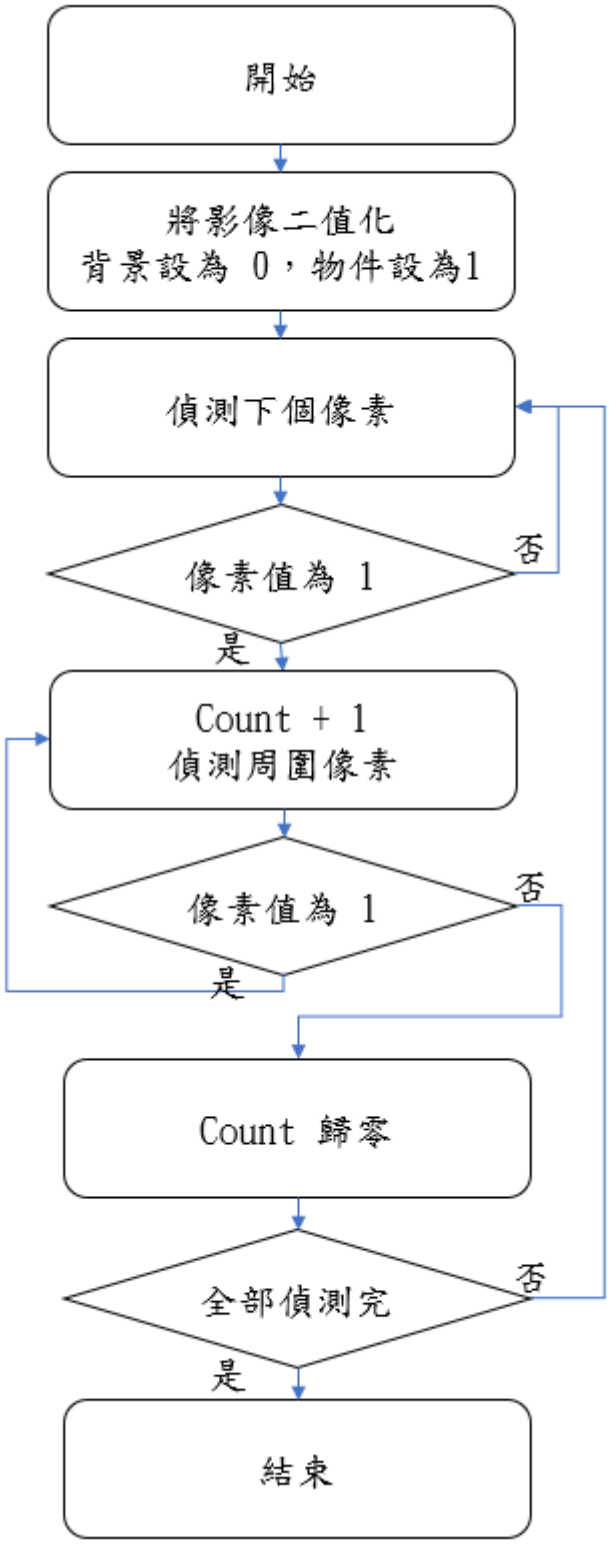


外流講義示意圖

```python
def main():

    test_name = 'test.png'
    ori_img = cv2.imread( test_name, cv2.IMREAD_GRAYSCALE )
    ori_img = cv2.copyMakeBorder( ori_img, 1, 1, 1, 1, cv2.BORDER_CONSTANT, value = 0 )
    print( ori_img.shape )
    cv2.imshow( 'Ori', ori_img )
    sys.setrecursionlimit( ori_img.shape[ 0 ] * ori_img.shape[ 1 ] )

    binary_img = Otsu( ori_img )                    # 二值化
    final_img, count = Count_pixel( binary_img )    # 跑遞迴，計算幾個連通域以及分別像素

    print( 'Image has', len( count ), 'components' )
    print( 'Each pixel is' )
    for i in count:
        print( i )

    cv2.waitKey()
    cv2.destroyAllWindows()
```

整體流程

1.2 演算法流程


開始
將影像二值化
背景設為 0，物件設為1
偵測下個像素
像素值為 1
否
是
Count + 1
偵測周圍像素
像素值為 1
否
是
Count 歸零
全部偵測完
否
是
結束

1.3 詳細流程與說明

1.3.1 二值化

本次圖片使用小畫家作圖，背景為白色，所以將較大的影像值設為 0，而物件設為 1

```
def Otsu( img ):
    ret_img = img.copy()
    row, col = img.shape
    for i in range( row ):
        for j in range( col ):
            if img[ i, j ] < 127:
                ret_img[ i, j ] = 1
            else:
                ret_img[ i, j ] = 0
    return ret_img
```

1.3.2 遞迴主函式

原影像全部掃一遍，若該像素值為 1 則進入遞迴往下尋找

```
def Count_pixel( img ):
    ret_img = img.copy()
    row, col = ret_img.shape
    label = 255
    count_list = []
    for i in range( 1, row - 1 ):
        for j in range( 1, col - 1 ):
            if ret_img[ i, j ] == 1:
                count = Dfs( ret_img, label, i, j )
                label -= 1
                count_list.append( count + 1 )
                ret_img[ i, j ] = 0
    return ret_img[ 1:row-1, 1:col-1 ], count_list
```

1.3.3 遞迴式

進入遞迴之後會以該點當出發點，繼續往下尋找為 1 的點

```
def Dfs( ret_img, label, r, c ):
    count = 0
    row, col = ret_img.shape
    ret_img[ r, c ] = label
    for next_r in range( r - 1, r + 2 ):
        for next_c in range( c - 1, c + 2 ):
            if ret_img[ next_r, next_c ] == 1:
                count += 1
                count += Dfs( ret_img, label, next_r, n
    return count
```

1.4 結果


```
Image has 6 components
Each pixel is
2348
1035
1332
1027
1346
898
```

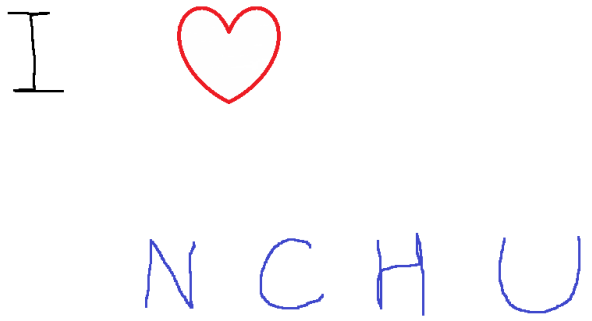
1.5 程式碼

```
import matplotlib.pyplot as plt
import numpy as np
import sys
import cv2

def Otsu( img ):
    ret_img = img.copy()
    row, col = img.shape
    for i in range( row ):
        for j in range( col ):
            if img[ i, j ] < 127:
                ret_img[ i, j ] = 1
            else:
                ret_img[ i, j ] = 0
    return ret_img

def Dfs( ret_img, label, r, c ):
    count = 0
    row, col = ret_img.shape
    ret_img[ r, c ] = label
    for next_r in range( r - 1, r + 2 ):
        for next_c in range( c - 1, c + 2 ):
            if ret_img[ next_r, next_c ] == 1:
                count += 1
                count += Dfs( ret_img, label, next_r, next_c )
    return count

def Count_pixel( img ):
    ret_img = img.copy()
    row, col = ret_img.shape
    label = 255
    count_list = []
    for i in range( 1, row - 1 ):
        for j in range( 1, col - 1 ):
            if ret_img[ i, j ] == 1:
                count = Dfs( ret_img, label, i, j )
```

```
                    label -= 1
                    count_list.append( count + 1 )
                    ret_img[ i, j ] = 0
    return ret_img[ 1:row-1, 1:col-1 ], count_list

def main():

    test_name = 'test.png'
    ori_img = cv2.imread( test_name, cv2.IMREAD_GRAYSCALE )
    ori_img = cv2.copyMakeBorder( ori_img, 1, 1, 1, 1,
cv2.BORDER_CONSTANT, value = 0 )
    print( ori_img.shape )
    cv2.imshow( 'Ori', ori_img )
    sys.setrecursionlimit( ori_img.shape[ 0 ] * ori_img.shape[ 1 ] )

    binary_img = Otsu( ori_img )                    # 二值化
    final_img, count = Count_pixel( binary_img )    # 跑遞迴，計算
幾個連通域以及分別像素

    print( 'Image has', len( count ), 'components' )
    print( 'Each pixel is' )
    for i in count:
        print( i )
    cv2.waitKey()
    cv2.destroyAllWindows()

main()
```

1.6 心得分享

這次實驗中原本我用的圖是這張



但是因為紅色太亮了，所以在處理二值化的時候與背景融合了，愛心就會不見，所以後來就使用白底黑字的方式來呈現。

這次作業使用的是 DFS 的方式，在網路上有看到一種叫做 Two-pass 的演算法，這個說明很清楚。
https://www.youtube.com/watch?v=ticZclUYy88&ab_channel=AaronBecker