R10521516 廖浚評

CV homework7

Use python OpenCV

Thinning operator use (Zhang-Suen Thinning Algorithm)

(https://rosettacode.org/wiki/Zhang-Suen_thinning_algorithm)

偽代碼

Zhang-Suen thinning steps:

While points are deleted **do**

For all pixels *p(i,j)* **do**

if (a)$2 \le B(P1) \le 6$

(b) $A(P1) = 1$

(c) Apply oneof the following:

1. $P2 \times P4 \times P6 = 0$ in odd iterations
2. $P2 \times P4 \times P8 = 0$ in even iterations

(d) Apply oneof the following:

1. $P4 \times P6 \times P8 = 0$ in odd iterations
2. $P2 \times P6 \times P8 = 0$ in even iterations

then

Deletepixel *p(i,j)*

endif

end for

end while

```python
import numpy as np
import cv2
import matplotlib.pyplot as plt
def img_binarize(img_in ):
    shape = img_in.shape
    binimg = np.zeros(shape )
    for i in range(shape[0]):
        for j in range(shape[1]):

                if img_in[i][j] >= 128:
                    binimg[i][j] =1
                else:
                    binimg[i][j] = 0
    return binimg


def img_downsample(img_in):
    #Downsampling Lena from 512x512 to 64x64,
    img_down = np.zeros((64,64), dtype =int)
    for i in range(img_down.shape[0]):
        for j in range(img_down.shape[1]):
            img_down[i][j] = img_in[8*i][8*j]
    return img_down

def getNeighbors(x,y ,img_in ):
    return [getValue(img_in, x-1, y), getValue(img_in, x-1, y+1), getValue(img_in, x, y+1),       \
            getValue(img_in, x+1,y+1), getValue(img_in, x+1,y), getValue(img_in, x+1, y-1),     \
            getValue(img_in, x,y-1), getValue(img_in, x-1,y-1)]

def getValue(img_in , x ,y):
    if x >=img_in.shape[0] or x <0 or y >= img_in.shape[1] or y <0:
        return 0
    return img_in[x][y]
```

前半段跟作業 6 一樣 binarize , downsample ,getneighbor (利用 getvalue 來判斷是否處於邊界), getvalue

```python
def transitions(getNeibors):
    n = getNeibors +getNeibors[0:1]
    return sum((n1,n2) == (0,1)for n1 ,n2 in zip(n , n[1:]))
```

判斷從 P1 ～ P8 中出現 0～1 的累計次數，其中 0 表示背景，1 表示前景。

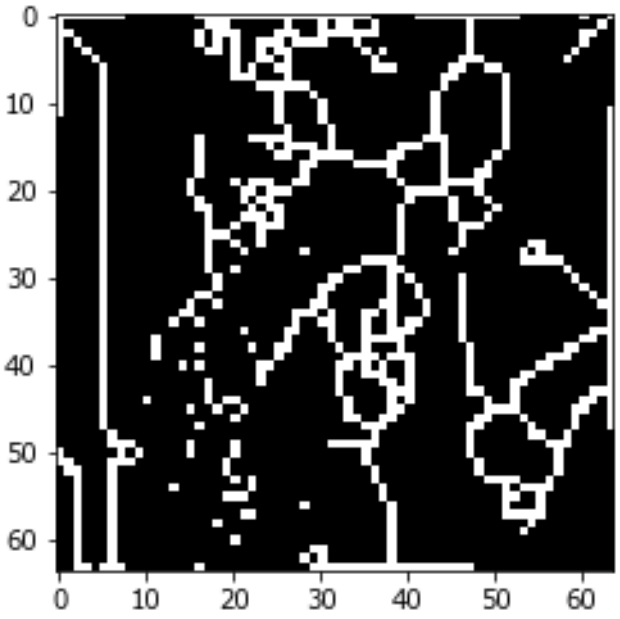
```
def thinning(image):
    changing1 = changing2 = [(-1, -1)]
    while changing1 or changing2:

        changing1 = []
        for x in range(1, len(image) - 1):
            for y in range(1, len(image[0]) - 1):
                P1,P2,P3,P4,P5,P6,P7,P8 = n = getNeighbors(x, y, image)
                if (image[x][y] == 1 and
                    P3 * P5 * P7 == 0 and
                    P1 * P3 * P5 == 0 and
                    transitions(n) == 1 and
                    2 <= sum(n) <= 6):
                    changing1.append((x,y))

        for x, y in changing1:
            image[x][y] = 0

        changing2 = []
        for x in range(1, len(image) - 1):
            for y in range(1, len(image[0]) - 1):
                P1,P2,P3,P4,P5,P6,P7,P8 = n = getNeighbors(x, y, image)
                if (image[x][y] == 1 and
                    P1 * P5 * P7 == 0 and
                    P1 * P3 * P7 == 0 and
                    transitions(n) == 1 and
                    2 <= sum(n) <= 6):
                    changing2.append((x,y))

        for x, y in changing2:
            image[x][y] = 0

    return image
```

1. pixel 不是背景值
2. P1 * P3* P5= 0
3. P3 * P5 * P7 = 0
4. S(P1) = 1，鄰域 neighbors 中相鄰兩個 neighbors 出現 0→1 的次數
5. 2 <= N(P0) <=6，中心圖元 P0 周圍的 neighbors（二值中的 1）的個數在 2~6 之間；

1. pixel 不是背景值
2. P1 * P5* P7 = 0
3. P1 * P3 * P7 = 0
4. S(P1) = 1，鄰域 neighbors 中相鄰兩個 neighbors 出現 0→1 的次數
5. 2 <= N(P0) <=6，中心圖元 P0 周圍的 neighbors（二值中的 1）的個數在 2~6 之間；

符合以上條件的刪除

重複做到沒有改變之後停止



用 512 的跑出來是