Homework 7　　資工所碩一 R05922068 彭宇劭

1. Write a program to generate thinned image

source code: hw7.py

執行方式：python hw7.py

版本：Python 2.7.10

Output：lena_thinning.txt / lena_thinning.bmp

簡述：前三步驟同Hw6

1. 在main function呼叫不同 function
2. 首先先將圖片二值化（Threshold=128）
3. 將二值化後的圖片 downsampling

原始圖512*512 pixels，利用8*8的小區域做 downsampling 取左上角的pixel為代表，其後會產生64*64的縮小圖

```
def down_sampling(img, k_size):
    o_size = img.shape[0]
    n_size = o_size/k_size
    down_img = np.zeros((n_size, n_size), dtype=int)
    for i in range(n_size):
        for j in range(n_size):
            down_img[i][j] = 1 if img[i*k_size][j*k_size]==255 else 0
    return down_img
```

4. 進入while-loop開始做thining的三個步驟，直到收斂跳出while-loop
    1. Mark-Interior / Border-Pixel
    2. The pair relationship
    3. Connected shrink

    下面分別說明

```
while True:
    pre_count = pixel_count(lena)
    int_bor = interior_border(lena)
    mark_int_bor = pair_relation(int_bor)
    lena = shrink(mark_int_bor)
    post_count = pixel_count(lena)
    if pre_count==post_count:
        break
```

4.1. Mark-Interior/Border-Pixel

利用8-connected，看每個pixel周圍八個點，如果都有值（非0）則該點為interior pixel，並將該點設為1，反之為border pixel設為2

```
def interior_border(down_img):
    # 0: nothing
    # 1: interior (i)
    # 2: border   (b)
    size = down_img.shape[0]
    int_bor = np.zeros((size, size), dtype=int)
    for i in range(down_img.shape[0]):
        for j in range(down_img.shape[1]):
            if(down_img[i][j]!=0):
                int_bor[i][j] = 1
                for m in [-1, 0, 1]:
                    for n in [-1, 0, 1]:
                        if (i+m<0 or i+m>=down_img.shape[0] or j+n<0 or j+n>=down_img.shape[1]):
                            int_bor[i][j] = 2
                            break
                        elif down_img[i+m][j+n]!=1:
                            int_bor[i][j] = 2
            else:
                int_bor[i][j] = 0
    return int_bor
```

4.2. The pair relationship

此步驟要找出在interior pixel周圍的點，這邊也用8-connected，觀察每個interior pixel，將他周圍的border pixel做記號（這邊標示為3）

```
def pair_relation(int_bor):
    # 0: nothing
    # 1: interior      (i)
    # 2: border        (b)
    # 3: marked border (m)
    for i in range(int_bor.shape[0]):
        for j in range(int_bor.shape[1]):
            if int_bor[i][j]==1:
                for m in [-1, 0, 1]:
                    for n in [-1, 0, 1]:
                        if(int_bor[i+m][j+n]!=1):
                            int_bor[i+m][j+n]=3
    return int_bor
```

4.3. Connected shrink

觀察每個做記號的border pixel（標記為3的點），利用投影片上的公式，計算他的connected值，如果去掉該點會分出大於一個component則該點要保留，反之，去掉該點（設為0）

```
def shrink(mark_int_bor):
    for i in range(mark_int_bor.shape[0]):
        for j in range(mark_int_bor.shape[1]):
            if mark_int_bor[i][j]==3:
                mark_int_bor[i][j]=yokoi(mark_int_bor, i, j)
    return mark_int_bor
```

```
def yokoi(img, i, j):
    a1 = h_func(img, [i,j], [i,j+1], [i-1,j+1], [i-1,j])
    a2 = h_func(img, [i,j], [i-1,j], [i-1,j-1], [i,j-1])
    a3 = h_func(img, [i,j], [i,j-1], [i+1,j-1], [i+1,j])
    a4 = h_func(img, [i,j], [i+1,j], [i+1,j+1], [i,j+1])
    a  = a1+a2+a3+a4
    return 1 if a>1 else 0
```

```
def h_func(img, x1, x2, x3, x4):
    size = img.shape[0]
    if x2[0]>=size or x2[1]>=size or x2[0]<0 or x2[1]<0:
        return 0
    if (img[x2[0]][x2[1]]==0):
        return 0
    if x3[0]>=size or x3[1]>=size or x3[0]<0 or x3[1]<0:
        return 1
    if x4[0]>=size or x4[1]>=size or x4[0]<0 or x4[1]<0:
        return 1
    elif img[x3[0]][x3[1]]==0 or img[x4[0]][x4[1]]==0:
        return 1
    else:
        return 0
```

結果：