HW4

- **A discription of your homework**

  Programming language used: Python 2.7

  Library used: Numpy, PIL, Scipy.misc

- **Your parameters**

  i: row

  j: column

  tem: 用於儲存每個像素的灰階數值 0~255

  kernel: octogonal 3-5-5-5-3 kernel

  J: kernel [(0,-1),(0,0),(1,0)]

  K: kernel [(-1,0),(-1,1),(0,1)]

- **Function**

  bi: binarize

  dil: dilation

  com: complement

  hit: hit-and-miss

- **The algorithm you used**

1. Binarize

   Threshold: 128

   灰階像素 >= 128 → 白(1)

   灰階像素 < 128 → 黑(0)

2. Complement

   若該像素為物件(1)，則改為背景(0)。若該像素為背景(1)，則改為物件(0)。

3. Dilation

   若該像素為物件，把所有 kernel 範圍裡的像素都變物件。

4. Erosion

   先取 complement 再和 kernel 的 inversion 做 dilation。

   Erosion-Dilation Duality:

   $$(A \ominus B)^c = A^c \oplus \check{B}$$

   $\check{B}$: symmetrical set of B with respect to origin

   inversion symmetry: $(\mathrm{x}, \mathrm{y}) \rightarrow (-\mathrm{x}, -\mathrm{y})$

   transpose of B

   由於 octogonal 3-5-5-5-3 kernel 有 inversion symmetry，所以$\check{B} = B$。

5. Opening

   $$\mathrm{B} \circ \mathrm{K} = (\mathrm{B} \ominus \mathrm{K}) \oplus \mathrm{K}$$

   先做 erosion，再做 dilation：把 erosion 後的結果丟到 dilation 跑一遍。

6. Closing

   $$\mathrm{B} \cdot \mathrm{K} = (\mathrm{B} \oplus \mathrm{K}) \ominus \mathrm{K}$$

先做 dilation，再做 erosion：把 dilation 後的結果丟到 erosion 跑一遍。

7. Hit-and-miss

$$A \otimes (J, K) = (A \ominus J) \cap (A^c \ominus K)$$

這裡的 erosion 作法和 2.不一樣。作法如下：

kernel 範圍裡的像素都是物件的話，將原點變成物件，否則為背景。

只要 kernel 範圍裡任一項素為背景，則跳出檢查迴圈。若通過整個檢查迴圈(即 kernel 範為裡所有像素皆為物件)，則將原點變成物件。

(1) $A \ominus J$

用 J 這個 kernel 來做 erosion。

(2) $A^c \ominus K$

先將原圖取 complement，再用 K 這個 kernel 來做 erosion。

(3) 取交集

- **Principal code fragment**

```
def bi(x):
    tem = np.zeros(x.shape)
    for i in range(x.shape[0]):
        for j in range(x.shape[1]):
            if x[i][j]<128:
                tem[i][j] = 0
            else:
                tem[i][j] = 1
    return tem

def com(x):
    tem = np.zeros(x.shape)
    for i in range(x.shape[0]):
        for j in range(x.shape[1]):
            if x[i][j]==0:
                tem[i][j]=1
    return tem

kernel = []
for k in range(-2,3,1):
    for m in range(-2,3,1):
        if (k!=-2 or m!=-2) and (k!=-2 or m!=2) and (k!=2 or m!=-2) and (k!=2 or m!=2):
            kernel.append([k,m])

def dil(x):
    tem = np.zeros(x.shape)
    for i in range(2,x.shape[0]-2,1):
        for j in range(2,x.shape[1]-2,1):
            if x[i][j]==1:
                for k in range(len(kernel)):
                    tem[i+kernel[k][0]][j+kernel[k][1]] = 1
    return tem
```

```
J = [(0,-1),(0,0),(1,0)]
K = [(-1,0),(-1,1),(0,1)]
def hit(x):
    y = com(x)
    tem_1 = np.zeros(x.shape)
    tem_2 = np.zeros(y.shape)
    tem = np.zeros(x.shape)
    for i in range(0,x.shape[0]-1,1):
        for j in range(1,x.shape[1],1):
            flag = 1
            for k in range(len(J)):
                    if x[i+J[k][0]][j+J[k][1]] == 0:
                            flag = 0
                            break
            if flag != 0:
                    tem_1[i][j] = 1

    for i in range(1,y.shape[0],1):
        for j in range(0,y.shape[1]-1,1):
            flag = 1
            for k in range(len(K)):
                    if y[i+K[k][0]][j+K[k][1]] == 0:
                            flag = 0
                            break
            if flag != 0:
                    tem_2[i][j] = 1

    for i in range(x.shape[0]):
        for j in range(x.shape[1]):
            if tem_1[i][j]==1 and tem_2[i][j]==1:
                tem[i][j]=1
    return tem

ans = bi(lena)
scipy.misc.imsave('lena_bi.jpg', ans)

ans_dil = dil(ans)
scipy.misc.imsave('lena_dil.jpg', ans_dil)

ans_ero = com(dil(com(ans)))
scipy.misc.imsave('lena_ero.jpg', ans_ero)

ans_opn = dil(ans_ero)
scipy.misc.imsave('lena_opn.jpg', ans_opn)

ans_clo = com(dil(com(ans_dil)))
scipy.misc.imsave('lena_clo.jpg', ans_clo)

ans_hit = hit(ans)
scipy.misc.imsave('lena_hit.jpg', ans_hit)
```

- **Resulting images**

1. Dilation

2. Erosion

3. Opening

4. Closing

5. Hit-and-miss