# HW4 Report

語言環境: python3 + PIL　　執行: python3 hw4.py

檔案:

　　程式 ------ hw4.py　morph.py

　　圖檔 ------ binarized.bmp　dilation.bmp　erosion.bmp

　　　　　　　opening.bmp　closing.bmp　rightUp.bmp

**hw4.py:**

```python
from PIL import Image
import numpy
import morph

im = Image.open('binarized.bmp')
(width,height), data_array = im.size, numpy.array(im)
pic = morph.PIC.imToPic(data_array)
Image.fromarray(pic.dilation(morph.tffft).toDataArray(), 'L').save('dilation.bmp')
print('dilation completed!')
Image.fromarray(pic.erosion(morph.tffft).toDataArray(), 'L').save('erosion.bmp')
print('erosion completed!')
Image.fromarray(pic.opening(morph.tffft).toDataArray(), 'L').save('opening.bmp')
print('opening completed!')
Image.fromarray(pic.closing(morph.tffft).toDataArray(), 'L').save('closing.bmp')
print('closing completed!')
Image.fromarray(pic.hitAndMiss(morph.J, morph.K).toDataArray(), 'L').save('rightUp.bmp')
```

　　將 binarized.bmp 讀入，並以此資料陣列創建一 morph.PIC 物件 pic

　　再用 morph.PIC 的成員函式作出對應的新 PIC 物件

　　最後用.toDataArray()成員函式得到其資料陣列

　　並用 Image.fromarray 和 save 存檔

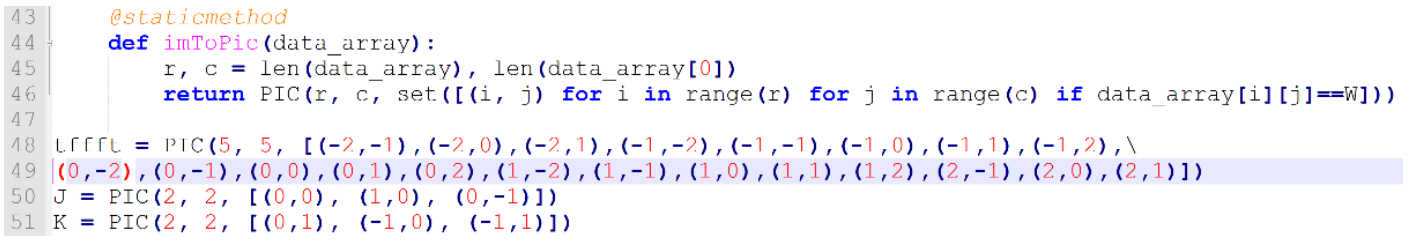
**morph.py:**

```python
class PIC:
    def __init__(self, row, column, points, compl=None):
        self.row, self.col = row, column
        self.Ws, self.Bs = points, compl
    def toDataArray(self):
        ret = numpy.zeros((self.row, self.col), dtype=numpy.uint8)
        for r,c in self.Ws: ret[r][c] = W
        return ret
    def getmmMM(self):
        mr, mc, Mr, Mc = None, None, None, None
        for i,j in self.Ws:
            if mr==None or mr>i: mr = i
            if mc==None or mc>j: mc = j
            if Mr==None or Mr<i: Mr = i
            if Mc==None or Mc<j: Mc = j
        return mr,mc,Mr,Mc
```

PIC 物件有 row 和 col 存放此圖片之大小

Ws 和 Bs 皆是 set 類別，各存放一些(r,c)表示為白色或黑色的點座標

getmmMM 返回白色點座標中的(最小 row, 最小 col, 最大 row, 最大 col)



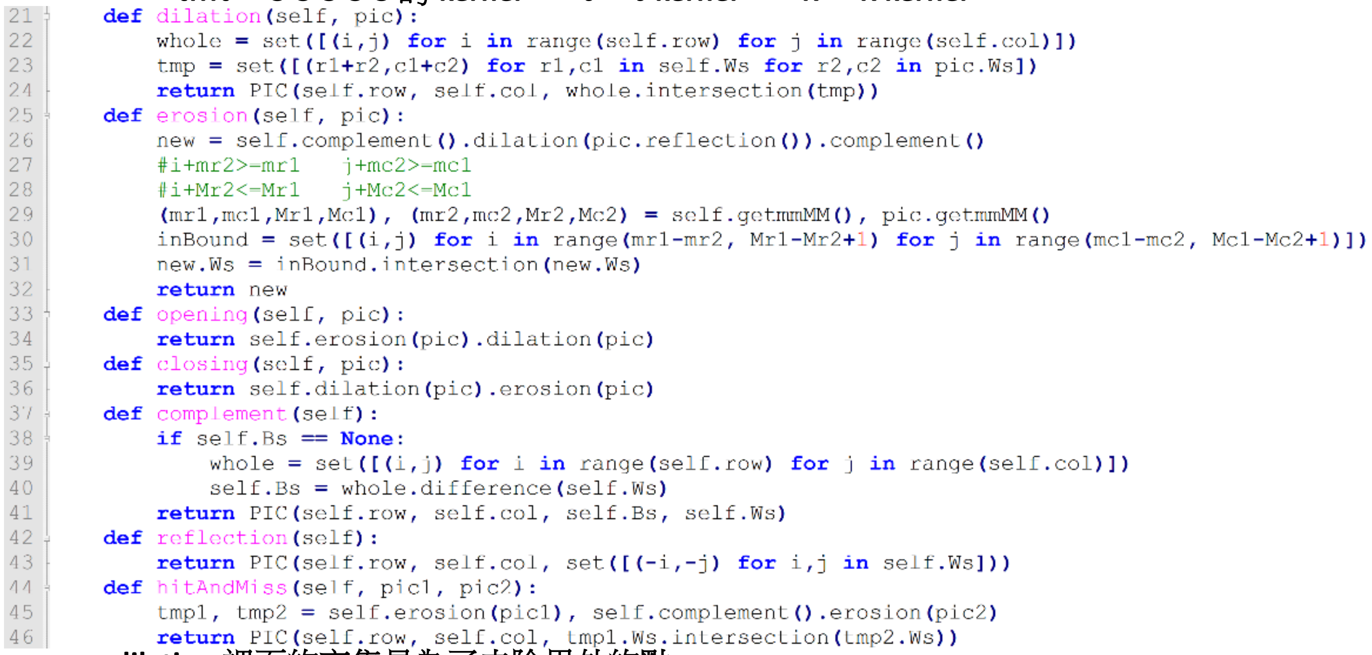
```
    @staticmethod
    def imToPic(data_array):
        r, c = len(data_array), len(data_array[0])
        return PIC(r, c, set([(i, j) for i in range(r) for j in range(c) if data_array[i][j]==W]))

tffft = PIC(5, 5, [(-2,-1),(-2,0),(-2,1),(-1,-2),(-1,-1),(-1,0),(-1,1),(-1,2),\
(0,-2),(0,-1),(0,0),(0,1),(0,2),(1,-2),(1,-1),(1,0),(1,1),(1,2),(2,-1),(2,0),(2,1)])
J = PIC(2, 2, [(0,0), (1,0), (0,-1)])
K = PIC(2, 2, [(0,1), (-1,0), (-1,1)])
```

**imToPic** 接收影像資料陣列並回傳對應的 **PIC** 物件

此外 **morph.py** 定義了三個全域的 **PIC** 物件:

**tffft -> 3-5-5-5-3** 的 **kernel**　　**J -> J kernel**　　**K -> K kernel**



```
    def dilation(self, pic):
        whole = set([(i,j) for i in range(self.row) for j in range(self.col)])
        tmp = set([(r1+r2,c1+c2) for r1,c1 in self.Ws for r2,c2 in pic.Ws])
        return PIC(self.row, self.col, whole.intersection(tmp))
    def erosion(self, pic):
        new = self.complement().dilation(pic.reflection()).complement()
        #i+mr2>=mr1    j+mc2>=mc1
        #i+Mr2<=Mr1    j+Mc2<=Mc1
        (mr1,mc1,Mr1,Mc1), (mr2,mc2,Mr2,Mc2) = self.getmmMM(), pic.getmmMM()
        inBound = set([(i,j) for i in range(mr1-mr2, Mr1-Mr2+1) for j in range(mc1-mc2, Mc1-Mc2+1)])
        new.Ws = inBound.intersection(new.Ws)
        return new
    def opening(self, pic):
        return self.erosion(pic).dilation(pic)
    def closing(self, pic):
        return self.dilation(pic).erosion(pic)
    def complement(self):
        if self.Bs == None:
            whole = set([(i,j) for i in range(self.row) for j in range(self.col)])
            self.Bs = whole.difference(self.Ws)
        return PIC(self.row, self.col, self.Bs, self.Ws)
    def reflection(self):
        return PIC(self.row, self.col, set([(-i,-j) for i,j in self.Ws]))
    def hitAndMiss(self, pic1, pic2):
        tmp1, tmp2 = self.erosion(pic1), self.complement().erosion(pic2)
        return PIC(self.row, self.col, tmp1.Ws.intersection(tmp2.Ws))
```

**dilation** 裡面的交集是為了去除界外的點

**erosion** 利用公式轉換為先取互補與對稱之膨脹再做互補，最後的交集是為了去除掉　-------因為界外而在 **dilation** 時被扣掉最後又在 **complement** 時補回來的點

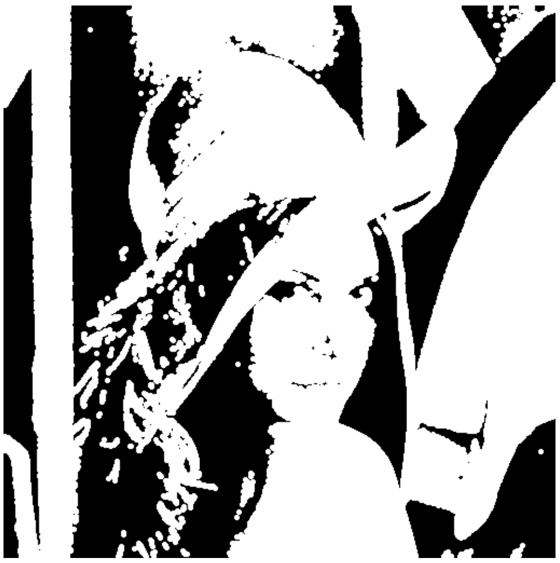
**complement** 先檢查是否已做過，若有則沿用之前存下的資料，若無則在規定的範圍內找尋不在原本 **Ws** 裡面的點

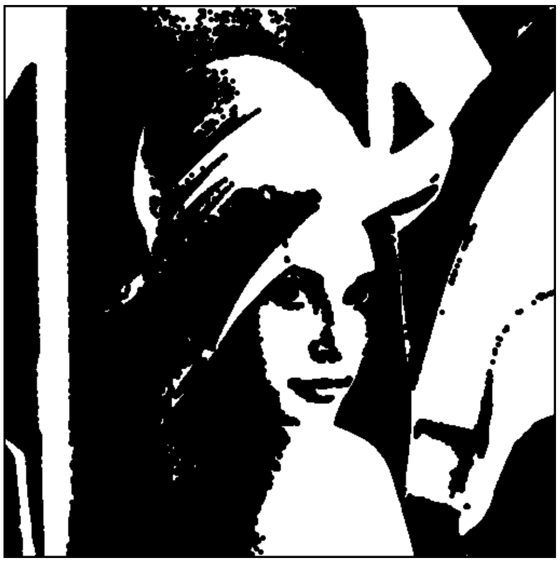
其他都是代公式

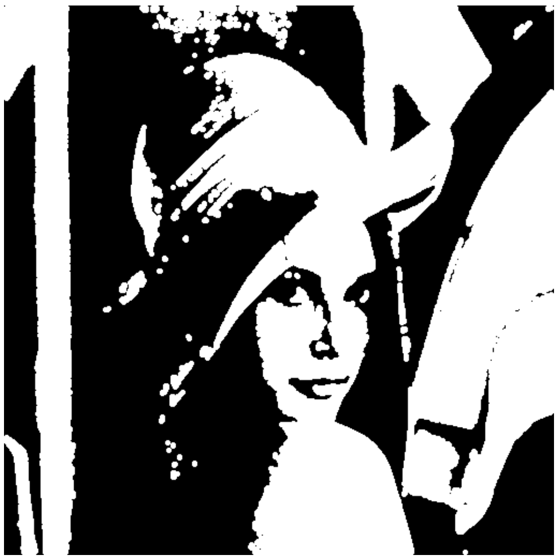
原圖:

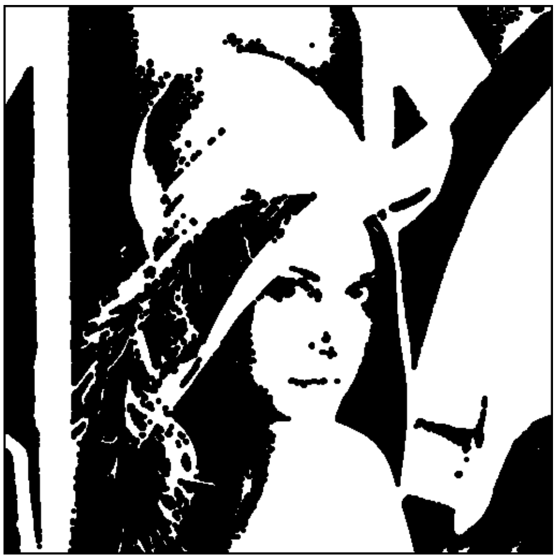
**binarized.bmp**

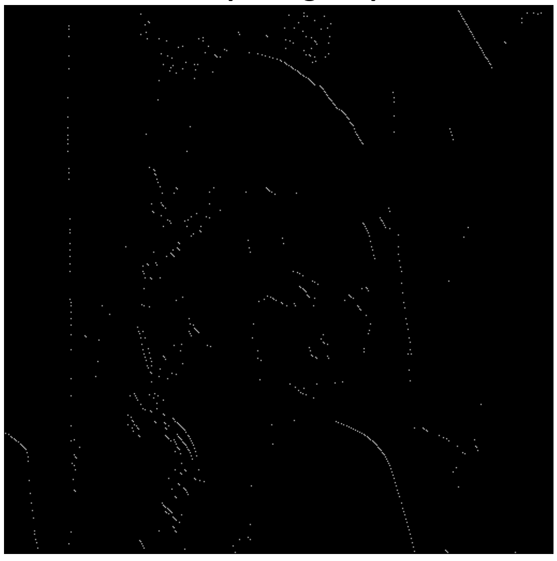
結果:


dilation.bmp


erosion.bmp


opening.bmp


closing.bmp


rightUp.bmp