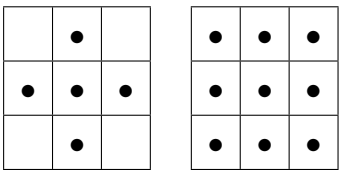
NTNU 影像處理 HW12

廖家緯

2020.6.4

- **Outline**:

  Implement the Lantuejoul's skeletonization method using the structuring element



Cross (B_1) Square (B_2)

  Apply the method to the following input image.

- **Code(Python):**

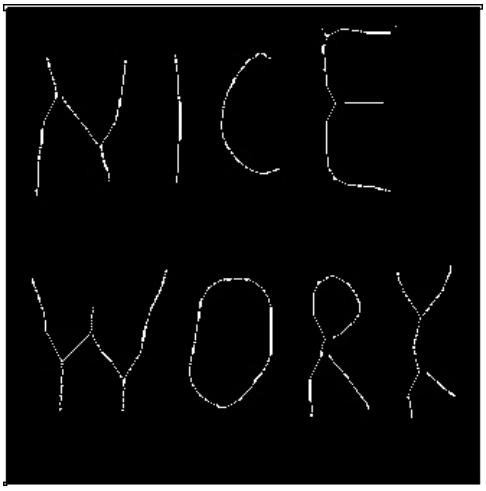
```python
# coding: utf-8
import numpy as np
import cv2

#轉成二值影像
def binary_img(I):
    I_ = I.copy()
    for i in range(n):
        for j in range(m):
            if I[i][j] > 128:
                I_[i][j] = 1
            else:
                I_[i][j] = 0
    return I_

#Lantuejoul' s method
def Lantuejouls_method(I, B):
    I_ = I.copy()
    Erosion = np.zeros((n, m)).astype('uint8')
    Opening = np.zeros((n, m)).astype('uint8')
    Differences = np.zeros((n, m)).astype('uint8')
    Union = np.zeros((n, m)).astype('uint8')
    k = 0

```

```
    while k < 50:
        for i in range(1, n-1):
            for j in range(1, m-1):
                if np.array_equal(I_[i-1:i+2, j-1:j+2]*B, B):
                    Erosion[i][j] = 1

        for i in range(1, n-1):
            for j in range(1, m-1):
                if Erosion[i][j] == 1:
                    Opening[i-1:i+2, j-1:j+2] = np.ones(3)

        Differences = I_ - Opening
        Union = Union + Differences
        k += 1
        if np.array_equal(Opening, np.zeros((n, m))):
            break
        else:
            I_ = Erosion
            Erosion = np.zeros((n, m)).astype('uint8')
            Opening = np.zeros((n, m)).astype('uint8')

    Union = (Union*255).astype('uint8')
    return Union

# 顯示圖片
def imgshow(img):
    cv2.imshow('My Image', img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

I = cv2.imread('img.png', cv2.IMREAD_GRAYSCALE)
n, m = np.shape(I)

B1 = np.array([[0, 1, 0], [1, 1, 1], [0, 1, 0]]).astype('uint8')
B2 = np.array([[1, 1, 1], [1, 1, 1], [1, 1, 1]])

I1 = binary_img(I)
I1 = Lantuejouls_method(I1, B1)
imgshow(I1)
cv2.imwrite('4-connected components.jpg', I1)

I2 = binary_img(I)
I2 = Lantuejouls_method(I2, B2)
imgshow(I2)
cv2.imwrite('8-connected components.jpg', I2)
```

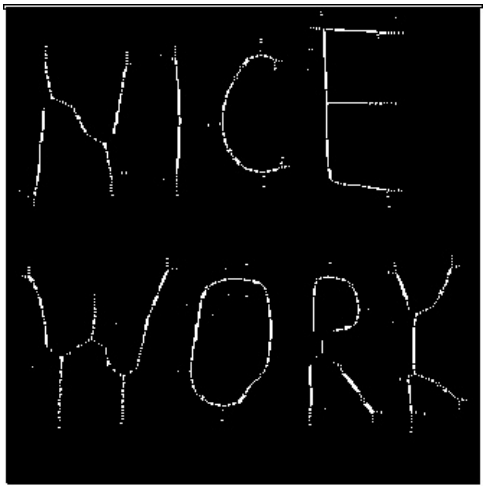
- **Input image:**

Input image

- **Result:**



4-connected components



8-connected components

- **Experience:**

這次作業一開始寫的 code 時間複雜度超過 $O(n^4)$，程式跑超久，後來和同學討論後砍掉重練，調整方法，並不斷優化，寫出這次作業。而我怕 k 值太大，導致程式執行時間超級久，因此我在迴圈上設了 $k < 50$，一般而言，執行差不多 50 次，已經可呈現效果。