CV Home Work 7 資研一 R07922003 劉濬慶

先將原圖變成 binary image: 512x512

- Step1: **Mark-Interior/Border-Pixel Operator**

$$h(c,d)=\begin{cases}c & if\ c=d\\ b & if\ c\neq d\end{cases}$$

$$f(c)=\begin{cases}b & if\ c=b\\ i & if\ c\neq b\end{cases}$$

- i : *interior label*
- b : *border label*

- $a_0 = x_0$
- $a_n = h(a_{n-1}, x_n)$

- *for 4-connected*

$a_n = h(a_{n-1}, x_n), n = 1,\dots,4$

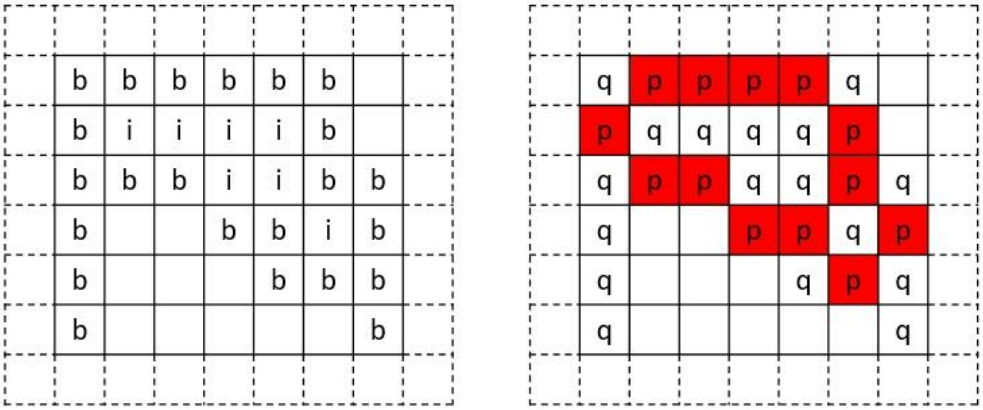
$output = f(a_4)$

- *for 8-connected*

$a_n = h(a_{n-1}, x_n), n = 1,\dots,8$

$output = f(a_8)$

- Step2: **Pair Relationship Operator**



for 4-connectivity

Let $l = b, m = i, \theta = 1$

$$h(a,i)=\begin{cases}1 & if\ a=i\\ 0 & otherwise\end{cases}$$

$$output=\begin{cases}q & if\ \sum_{n=1}^{4} h(x_n,i) < 1 \vee x_0 \neq b\\ p & if\ \sum_{n=1}^{4} h(x_n,i) \geq 1 \wedge x_0 = b\end{cases}$$

- Step3: **Connected Shrink Operator**

6.2.6 Connected Shrink Operator

- for 4-connectivity

$$h(b,c,d,e) = \begin{cases} 1 & if\ b = c \wedge (b \neq d \vee b \neq e) \\ 0 & otherwise \end{cases}$$

- for 8-connectivity

$$h(b,c,d,e) = \begin{cases} 1 & if\ b \neq c \wedge (b = d \vee b = e) \\ 0 & otherwise \end{cases}$$

Corner Neighborhood
(for corresponding x_i)

	x_2	x_6
		x_1

x_7	x_2	
x_3		

- $a_1 = h(x_0, x_1, x_6, x_2)$
- $a_1 = h(x_0, x_2, x_7, x_3)$
- $a_1 = h(x_0, x_3, x_8, x_4)$
- $a_1 = h(x_0, x_4, x_5, x_1)$

x_3		
x_8	x_4	

		x_1
	x_4	x_5

- $output = f(a_1, a_2, a_3, a_4, x_0) = \begin{cases} g & if\ exactly\ one\ of\ a_1, a_2, a_3, a_4 = 1 \\ x_0 & otherwise \end{cases}$

Yokoi Connectivity Number
: label 1(edge)

Scan: top-down left-right
Input: last-step output (not the original image)

- Step4: 比較 **Pair Relationship Matrix** and **Connected Shrink Matrix** 對 binary image 進行修改

- **Repeat 1~4 直到第 4 個 step binary image 沒有修改**

- 程式碼

```
from PIL import Image, ImageDraw

import numpy as np


def interior_border(matrix): # 1=border 2=interior

    i_b_matrix = np.zeros((514,514),dtype=int)

    for i in range(1,513):

        for j in range(1,513):

            x0 = matrix[i][j]

            x1 = matrix[i][j+1]

            x2 = matrix[i-1][j]

            x3 = matrix[i][j-1]

            x4 = matrix[i+1][j]

            #x5 = matrix[i+1][j+1]

            #x6 = matrix[i-1][j+1]

            #x7 = matrix[i-1][j-1]

            #x8 = matrix[i+1][j-1]


            if x0==255:
```

```
                    if x1==x2==x3==x4==255:
                        i_b_matrix[i][j] = 2
                    else:
                        i_b_matrix[i][j] = 1
    return i_b_matrix


def Pair(matrix): # 1=p 2=q
    Pair_matrix = np.zeros((514,514),dtype=int)
    for i in range(1,513):
        for j in range(1,513):
            x0 = matrix[i][j]
            x1 = matrix[i][j+1]
            x2 = matrix[i-1][j]
            x3 = matrix[i][j-1]
            x4 = matrix[i+1][j]

            if x0==1:
                if x1==2 or x2==2 or x3==2 or x4==2:
                    Pair_matrix[i][j]=1
```

```
        return Pair_matrix



def h3(b,c,d,e):
        if b==c and(b!=d or b!=e):
                return 1
        else:
                return 0


def f3(a1,a2,a3,a4,x0):
        if (a1+a2+a3+a4==1):
                return 1
        else:
                return 0


def Connected_Shrink(matrix):
        Connected_Shrink_matrix =
np.zeros((514,514),dtype=int)
        for i in range(1,513):
```

```
            for j in range(1,513):
                if matrix[i][j]==255:
                    x0 = matrix[i][j]
                    x1 = matrix[i][j+1]
                    x2 = matrix[i-1][j]
                    x3 = matrix[i][j-1]
                    x4 = matrix[i+1][j]
                    x5 = matrix[i+1][j+1]
                    x6 = matrix[i-1][j+1]
                    x7 = matrix[i-1][j-1]
                    x8 = matrix[i+1][j-1]

                    a1 = h3(x0, x1, x6, x2)
                    a2 = h3(x0, x2, x7, x3)
                    a3 = h3(x0, x3, x8, x4)
                    a4 = h3(x0, x4, x5, x1)
                    Connected_Shrink_matrix[i][j] =
f3(a1,a2,a3,a4,x0)
        return Connected_Shrink_matrix
```

```
'''
7 2 6
3 0 1
8 4 5
'''

lena=Image.open("lena.bmp")
matrix = np.array(lena)
pix=lena.load()
coulmn,row=lena.size
binary_matrix = np.zeros((514,514),dtype=int)
for i in range(512):
    for j in range(512):
        if matrix[i,j] < 128:
            binary_matrix[j+1][i+1] = 0
        else:
            binary_matrix[j+1][i+1] = 255

change=1
```

```
iter_cnt=1
while(change):
    print(iter_cnt)
    iter_cnt += 1
    change=0
    i_b_matrix = interior_border(binary_matrix)
    Pair_matrix = Pair(i_b_matrix)

Connected_Shrink_matrix=Connected_Shrink(binary_matrix)
    for i in range(1,513):
        for j in range(1,513):
            if (Pair_matrix[i][j]==1) and
(Connected_Shrink_matrix[i][j]==1) :
                binary_matrix[i][j]=0
                change=1

image=Image.new(lena.mode,(512,512))
for i in range(1,513):
    for j in range(1,513):
```

```
                if binary_matrix[i][j]==255:
                        image.putpixel((i-1,j-1),255)
image.save('thinning.bmp')
```