

CV hw7 / 電機所 R06921082 陳與賢

Description:

利用 python 來處理 bmp 檔，進行 **Thinning**

首先讀入 hw2 的 binary 圖檔，code 如下：

```
3
4 img_input = Image.open('binary.bmp')
5 pixels_input = img_input.load()
6
```

Algorithm:

首先要先做 marked-interior/border-pixel operator，依照投影片 8-connectivity 的作法標出是"b"（我記為 1000）還是"i"（我記為 2000），如下圖所示：

$$\begin{aligned} \bullet h(c, d) &= \begin{cases} c & \text{if } c = d \\ b & \text{if } c \neq d \end{cases} \\ \bullet f(c) &= \begin{cases} b & \text{if } c = b \\ i & \text{if } c \neq b \end{cases} \end{aligned}$$

$$\begin{aligned} \bullet & \text{for 8-connected} \\ & a_n = h(a_{n-1}, x_n), n = 1, \dots, 8 \\ & \text{output} = f(a_8) \end{aligned}$$

因為投影片沒有特別說明針對 boundary 的 pixel 要如何處理，因此我假設在 boundary 之 pixels 皆會被當作 border，也就是我一開始會給超出 boundary 的 pixels 給值 100（因為 binary 的輸入圖只會有 255 跟 0，所以掃到 100 的話，當前檢查的值就會被標成"b"），code 如下：

```

for y in range(512):
    for x in range(512):
        a = np.zeros(9)
        xx = np.zeros(9)
        a[0] = xx[0] = thinning_pixels[x, y]

        for i in range(1, 9):
            # impossible pixel value, i.e.: take boundary pixels as border
            xx[i] = 100

        if x + 1 < 512:
            xx[1] = thinning_pixels[x + 1, y]
        if x - 1 > -1:
            xx[3] = thinning_pixels[x - 1, y]
        if y + 1 < 512:
            xx[4] = thinning_pixels[x, y + 1]
        if y - 1 > -1:
            xx[2] = thinning_pixels[x, y - 1]
        if x + 1 < 512 and y + 1 < 512:
            xx[5] = thinning_pixels[x + 1, y + 1]
        if x + 1 < 512 and y - 1 > -1:
            xx[6] = thinning_pixels[x + 1, y - 1]
        if x - 1 > -1 and y - 1 > -1:
            xx[7] = thinning_pixels[x - 1, y - 1]
        if x - 1 > -1 and y + 1 < 512:
            xx[8] = thinning_pixels[x - 1, y + 1]

        for j in range(8):
            if a[j] == xx[j + 1]:
                a[j + 1] = a[j]
            else:
                a[j + 1] = 1000 # 1000 = "b"

        if a[8] == 1000:
            mark_arr[x, y] = 1000
        else:
            mark_arr[x, y] = 2000 # 2000 = "i"

```

再來要做 pair relationship operator，依據投影片 8-connectivity 的作法來 relabel，如下圖所示：

<p>Let $l = b, m = i, \theta = 1$</p> $h(a, i) = \begin{cases} 1 & \text{if } a = i \\ 0 & \text{otherwise} \end{cases}$	<p style="text-align: center;"><i>for 8-connected</i></p> $\text{output} = \begin{cases} q & \text{if } \sum_{n=1}^8 h(x_n, m) < \theta \vee x_0 \neq l \\ p & \text{if } \sum_{n=1}^8 h(x_n, m) \geq \theta \wedge x_0 = l \end{cases}$
---	--

這邊 threshold 的值因為作業沒有特別說明，所以我們設成跟投影片一樣 $\theta = 1$ ，白話一點也就是如果當前的 pixel 被標記為“b”（我記為 1000），而其八連通的 pixel 有任一個被標記為“i”（我

記為 2000) 的話，那當前的 pixel 被 relabel 為 p (我記為 3000)，code 如下：

```
for y in range(512):
    for x in range(512):
        if mark_arr[x, y] == 1000:
            if x + 1 < 512:
                if mark_arr[x + 1, y] == 2000:
                    mark_arr[x, y] = 3000 # 3000 = "p"
                    continue
            if x - 1 > -1:
                if mark_arr[x - 1, y] == 2000:
                    mark_arr[x, y] = 3000
                    continue
            if y + 1 < 512:
                if mark_arr[x, y + 1] == 2000:
                    mark_arr[x, y] = 3000
                    continue
            if y - 1 > -1:
                if mark_arr[x, y - 1] == 2000:
                    mark_arr[x, y] = 3000
                    continue
            if x + 1 < 512 and y + 1 < 512:
                if mark_arr[x + 1, y + 1] == 2000:
                    mark_arr[x, y] = 3000
                    continue
            if x + 1 < 512 and y - 1 > -1:
                if mark_arr[x + 1, y - 1] == 2000:
                    mark_arr[x, y] = 3000
                    continue
            if x - 1 > -1 and y - 1 > -1:
                if mark_arr[x - 1, y - 1] == 2000:
                    mark_arr[x, y] = 3000
                    continue
            if x - 1 > -1 and y + 1 < 512:
                if mark_arr[x - 1, y + 1] == 2000:
                    mark_arr[x, y] = 3000
                    continue
```

再來進行最後的部分，也就是 connected shrink operator，依據公式會發現其實就是 yokoi number = 1 的時候，因此我們從圖的左上到右下，依序對每個 pixel 值執行 hw6 的演算法找出 yokoi number，若 yokoi number = 1 且相應的 mark 為 "p" (我記為

3000)，則當前 pixel 值改成 0。另外此演算法需不斷迭代直到沒有改變為止，code 如下：

```
51
52 change = 1
53 while change == 1:
54     change = 0
55     mark()
56     for y in range(512):
57         for x in range(512):
58             if thining_pixels[x, y] == 255:
59                 if find_number(x, y) == 1 and mark_arr[x, y] == 3000:
60                     thining_pixels[x, y] = 0
61                     change = 1
62
```

另外 marked-interior/border-pixel operator 跟 pair relationship operator 我都是用 8-connectivity，yokoi number 也用 8-connectivity 的話 result 跟助教給的解答差蠻多的...，所以也附上 yokoi number 用 4-connectivity 的 result，跟助教給的解答較相近。

所以我 code 會產生兩個圖，一個 yokoi number 是 4-connectivity，另一個是 8-connectivity

（marked-interior/border-pixel operator 跟 pair relationship operator 都是 8-connectivity，因為我看投影片第 78 頁有標 8-connectivity）

Result:

Yokoi number 4-connectivity



Yokoi number 4-connectivity

