CV HW7

R11922196 林佑鑫

(a) **Thinning**

Brief description, algorithm:

Step 1: 用 HW1 的 shrink 跟 HW2 的 binarize function 做 down-sampling。

Step 2: 依照講義 6.2.5.1 的公式實作 Yokoi h_function 與 f_function，用 Yokoi connectivity number 換算 border/interior pixels。

Step 3: 依照講義的公式實作 Pair relationship h_function 與 f_function，用來換作 marked image。

Step 4: 依照講義的公式實作 connected shrink h_function 與 f_function，用來刪除不必要的非連接點。

Parameters: None

Principal code fragment:

Step 1:

```python
def binarize(img):
    height, width = img.shape
    binarize_img = np.zeros_like(img)

    for h in range(height):
        for w in range(width):
            if((img[h, w] >= 128)):
                binarize_img[h, w] = 255
            else:
                binarize_img[h, w] = 0
    return binarize_img

def shrink(img, ratio):
    height, width = img.shape
    new_height = height // ratio
    new_width = width // ratio
    shrink_img = np.zeros((new_height, new_width))

    for h in range(new_height):
        for w in range(new_width):
            shrink_img[h, w] = img[h * ratio, w * ratio]

    return shrink_img
```

Step 2:

```python
def yokoi_connectivity(img):
    height, width = img.shape

    pad_img = np.zeros((height+2, width+2)) - 1
    pad_img[1:height+1, 1:width+1] = img

    output_img = np.zeros_like(img) - 1

    for h in range(1, height+1):
        for w in range(1, width+1):
            if pad_img[h, w] == 255:
                a1 = h_function(pad_img[h, w], pad_img[h, w+1], pad_img[h-1, w+1], pad_img[h-1, w])
                a2 = h_function(pad_img[h, w], pad_img[h-1, w], pad_img[h-1, w-1], pad_img[h, w-1])
                a3 = h_function(pad_img[h, w], pad_img[h, w-1], pad_img[h+1, w-1], pad_img[h+1, w])
                a4 = h_function(pad_img[h, w], pad_img[h+1, w], pad_img[h+1, w+1], pad_img[h, w+1])

                output_img[h-1, w-1] = f_function(a1, a2, a3, a4)

    return output_img

def h_function(b, c, d, e):
    if b == c and (b != d or b != e):
        return 'q'
    elif b == c == d == e:
        return 'r'
    else:
        return 's'

def f_function(a1, a2, a3, a4):
    if a1 == a2 == a3 == a4 == 'r':
        return 5
    else:
        return [a1, a2, a3, a4].count('q')
```

Step 3:

```python
def pair_relationship(img):
    height, width = img.shape

    pad_img = np.zeros((height+2, width+2))
    pad_img[1:height+1, 1:width+1] = img

    output_img = np.zeros((height, width))

    for h in range(1, height+1):
        for w in range(1, width+1):
            x7, x2, x6 = pad_img[h-1, w-1], pad_img[h-1, w], pad_img[h-1, w+1]
            x3, x0, x1 = pad_img[h, w-1], pad_img[h, w], pad_img[h, w+1]
            x8, x4, x5 = pad_img[h+1, w-1], pad_img[h+1, w], pad_img[h+1, w+1]
            if x0 == -1:
                output_img[h-1, w-1] = 0
            else:
                output_img[h-1, w-1] = pair_f_function(x0, x1, x2, x3, x4)

    return output_img

def pair_h_function(a, m):
    if a == m:
        return 1
    else:
        return 0

def pair_f_function(x0, x1, x2, x3, x4):
    m = 1
    if (pair_h_function(x1, m) + pair_h_function(x2, m) + pair_h_function(x3, m) + pair_h_function(x4, m)) >= 1 and x0 == m:
        return 1
    else:
        return 2
```

Step 4:

```python
def connected_shrink(img):
    height, width = img.shape

    tmp_img = np.zeros((height, width))
    tmp_img[img > 0] = 1
    pad_img = np.zeros((height+2, width+2))
    pad_img[1:height+1, 1:width+1] = tmp_img

    for h in range(1, height+1):
        for w in range(1, width+1):
            if img[h-1, w-1] == 1:
                x7, x2, x6 = pad_img[h-1, w-1], pad_img[h-1, w], pad_img[h-1, w+1]
                x3, x0, x1 = pad_img[h, w-1], pad_img[h, w], pad_img[h, w+1]
                x8, x4, x5 = pad_img[h+1, w-1], pad_img[h+1, w], pad_img[h+1, w+1]

                a1 = connected_h_function(x0, x1, x6, x2)
                a2 = connected_h_function(x0, x2, x7, x3)
                a3 = connected_h_function(x0, x3, x8, x4)
                a4 = connected_h_function(x0, x4, x5, x1)

                if connected_f_function(a1, a2, a3, a4, 1) == 0:
                    img[h-1, w-1] = 0
                    pad_img[h, w] = 0

            if img[h-1, w-1] != 0:
                img[h-1, w-1] = 255

    return img

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

def connected_f_function(a1, a2, a3, a4, x):
    if [a1, a2, a3, a4].count(1) == 1:
        return 0
    else:
        return x
```

Resulting image:

Thinning 1: 

Thinning 2: 

Thinning 3: 

Thinning 4: 

Thinning 5: 

Thinning 6: