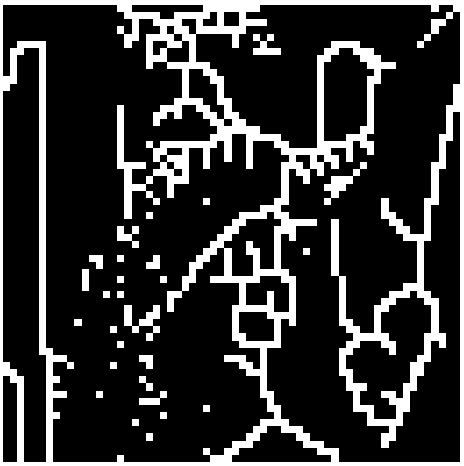
# 2021 CV HW7 Report

## Results



## Thinning Operator

**Step 1.** 對 512 x 512 的原圖做 thresholding 再做 down sampling 成 64 x 64，用 8 x 8 的區塊為單位取最左上角的 pixel value 做為採樣點。

```
threshold, image = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)
# down sample
downsample = np.zeros((h//8, w//8), np.uint8)
for i in range(0, h, 8):
    for j in range(0, w, 8):
        downsample[i//8][j//8] = img[i][j]
```

**Step 2.** 依照 Yokoi Connectivity Number 的公式去實作 function。

$$h(b,c,d,e) = \begin{cases} q & if\ b = c\ and\ (d \neq b\ \vee e \neq b) \\ r & if\ b = c\ and\ (d = b \wedge e = b) \\ s & if\ b \neq c \end{cases}$$

$$f(a_1,a_2,a_3,a_4) = \begin{cases} 5 & if\ a_1 = a_2 = a_3 = a_4 = r \\ n & where\ n = numberof\{a_k | a_k = q\}, otherwise \end{cases}$$

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

def f(a1, a2, a3, a4):
```

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

**Step 3.** 若 pixel value 為 0，label 則為 0；若不為 0，則去計算 a1、a2、a3、a4（如下圖），再將 a1 ~ a4 帶入 f 函式中，得到 Yokoi connectivity number 的 output。

$$a_1 = h(x_0, x_1, x_6, x_2) \quad a_3 = h(x_0, x_3, x_8, x_4)$$
$$a_2 = h(x_0, x_2, x_7, x_3) \quad a_4 = h(x_0, x_4, x_5, x_1)$$
$$output = f(a_1, a_2, a_3, a_4)$$

```
def Yokoi():
    for i in range(hh):
        for j in range(ww):
            if downsample[i][j] == 0:
                yokoi[i][j] = 0
                continue
            # find x0 ~ x8's pixel value
            x = pixelValues(downsample, i, j)
            a1 = h(x[0], x[1], x[6], x[2])
            a2 = h(x[0], x[2], x[7], x[3])
            a3 = h(x[0], x[3], x[8], x[4])
            a4 = h(x[0], x[4], x[5], x[1])
            yokoi[i][j] = f(a1, a2, a3, a4)
```

**Step 4.** 依照以下 Pair Relationship Operator 的公式去實作 function。若 Yokoi 的 pixel value 為 1 且周圍的 4 個鄰居有一個以上也為 1 則標為 ‘p’ ，其餘則為 ‘q’ 。

H function: (m="1", means "edge" in Yokoi)

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

Output:

- $y = \begin{cases} q, if\ \sum_{n=1}^{4} h(x_n, m) < 1\ or\ x_0 \neq m \\ p, if\ \sum_{n=1}^{4} h(x_n, m) \geq 1\ and\ x_0 = m \end{cases}$

```
# Pair relationship operator
def H(a, m):
    if a == m:
        return 1
    return 0

def y(x, m):
    sum = 0
    for i in range(1, 5):
        sum += H(x[i], m)
    if sum < 1 or x[0] != m:
```

```
        return 'q'
    elif sum >= 1 and x[0] == m:
        return 'p'
    return ''

def pairOperator():
    for i in range(hh):
        for j in range(ww):
            if yokoi[i][j] == 0:
                continue
            x = pixelValues(yokoi, i, j) # find x0 ~ x8's pixel value
            pair[i][j] = y(x, 1)
```

**Step 5.** 依照以下 Connected Shrink Operator 的公式去實作 function。

H function: (yokoi corner => "q")

- $h(b,c,d,e) = \begin{cases} 1, if\ b = c\ and\ (d \neq b\ or\ e \neq b) \\ 0, otherwise \end{cases}$

Output:

- $f(a_1,a_2,a_3,a_4,x) = \begin{cases} g, if\ exactly\ one\ of\ a_n = 1, n = 1\sim4 \\ x, otherwise \end{cases}$

```
# Connected Shrink Operator
def h_connected(b, c, d, e):
    if b == c and (d != b or e !=b):
        return 1
    return 0

def f_connected(a1, a2, a3, a4, x0):
    if sum([a1, a2, a3, a4]) == 1:
        return 0 # background
    return x0
```

**Step 6.** 有了以上 3 個 Operator 的 function 後，就可以來實作 Thinning Operator 了。先對圖片做 Yokoi Operator 和 Pair Relationship Operator 產生 marked image。接著對 Pair Relationship Operator 後是 'p' 的 pixel value 去做 Connected Shrink Operator。

```
# Thinning Operator
def thinning():
    Yokoi()
    pairOperator()
    for i in range(hh):
        for j in range(ww):
            if pair[i][j] != 'p':
                continue
            x = pixelValues(downsample, i, j)
```

```
            a1 = h_connected(x[0], x[1], x[6], x[2])
            a2 = h_connected(x[0], x[2], x[7], x[3])
            a3 = h_connected(x[0], x[3], x[8], x[4])
            a4 = h_connected(x[0], x[4], x[5], x[1])
            downsample[i][j] = f_connected(a1, a2, a3, a4, x[0])
```

**Step 7.** 對 down sample 完的照片做 7 次 iteration（Thinning Operator），即得到最終的 output。

```
# 7 iteration
for i in range(7):
    thinning()
```