# Homework 7

Thinning

R09922063 鄭筠庭 資工所碩一

Programming language: Python 3.7.3

Library used for this homework:

- ✦ Numpy: for array usage
- ✦ Opencv: to read and write the image file

Image info: lena.bmp [512(width),512(height),1(Grayscale)]

## Thinning Operation:

- ➢ Create marked image
  1. Yokoi Operator
  2. Pair Relationship Operator
- ➢ Connected Shrink Operator
- ➢ Compare the shrink result with marked image

### - Pair Relationship Operator:

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

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

➢ Output:

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

### - Connected Shrink Operator:

➢ 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:

- $f(a_1, a_2, a_3, a_4, x) = \begin{cases} g, if\ exactly\ one\ of\ a_n = 1, n = 1{\sim}4 \\ x, otherwise \end{cases}$

## Code explanation:

The first few step is the same as homework 6. Binarize the benchmark image lena.bmp as in HW2, and down sampling the binary image from a 512x512 to 64x64.

```
###--- Binarize the benchmark image lena ---###
def binary(image):
    for r in range(rows):
        for c in range(columns):
            if image[r,c] < 128:
                image[r,c] = 0
            else:
                image[r,c] = 255
    return image

###--- Downsampling Lena from 512x512 to 64x64 ---###
def downsampling(image, divide, smallImage):
    for r in range(0, rows, divide):
        for c in range(0, columns, divide):
            smallImage[int(r/divide), int(c/divide)] = image[r,c]
```

In the main function, do the Yokoi Operation and Pair Relationship Operation. Take the marked Lena image and the original image to the Thinning operator. Repeat these steps until the original image and marked image are the same.

```
markLena = Yokoi(lena)
markLena = thinning(lena, markLena)

while not np.array_equal(lena, markLena):
    lena = markLena
    markLena = Yokoi(lena)
    markLena = thinning(lena, markLena)
```

**Yokoi function** is the same as the previous homework. The image will be labeled from 0 to 5. Use this labeled image to do the pair relationship operation.

```
qNum = 0
sNum = 0
for h in range(1,5): # 4-connected
    if copyImage[r+1,c+1] != copyImage[r+1+mask[h][0], c+1+mask[h][1]]:
        sNum += 1
    if (copyImage[r+1,c+1] == copyImage[r+1+mask[h][0], c+1+mask[h][1]]) and (copyImage[r+1,c+1] !=
        copyImage[r+1+mask[h+1][0], c+1+mask[h+1][1]] or copyImage[r+1,c+1] != copyImage[r+1+mask[h+5][0]
        , c+1+mask[h+5][1]]):
        qNum += 1

if qNum == 4:
    resultImage[r,c] = 4
elif qNum == 3:
    resultImage[r,c] = 3
elif qNum == 2:
    resultImage[r,c] = 2
elif qNum == 1:
    resultImage[r,c] = 1
elif qNum == 0 and sNum == 0:
    resultImage[r,c] = 5
```

**Pair relationship operator** finds the pixel that is labeled as 1, and check if its neighbors have a least one is labeled as 1 as well. If the conditions are satisfied, we mark this pixel as a “p” pixel.

```
###--- pair relationship operator ---###
def pairRelation(YokoiImage):
    for r in range(rows):
        for c in range(columns):
            if YokoiImage[r,c] != 1:
                continue

            for num in range(1,5):
                if (r+mask[num][0]>=0) and (r+mask[num][0]<rows) and (c+mask[num][1]>=0) and (c+mask[num][1]<columns):
                    if YokoiImage[r+mask[num][0], c+mask[num][1]] == 1 or YokoiImage[r+mask[num][0], c+mask[num][1]]
                        == 9:
                        YokoiImage[r,c] = 9 # Mark as 'p'(Set value = 9)
                        break

    return YokoiImage
```

**Thinning operation** finds the ‘p’ marking pixel and check if it is an edge pixel in Yokoi operation. Since this is a recursive operation, the pixels will be deleted immediately after it was found. As a result, not all the ‘p’ marking pixel will be deleted.

```
for r in range(rows):
    for c in range(columns):
        # Only do thinning on the 'p' pixel
        if markedImage[r,c] != 9:
            continue

        # Do the Yokoi operation
        qNum = 0
        for h in range(1,5): # 4-connected
            # Check the 4 neighbors and count how many 'q' does this pixel get
            if (copyImage[r+1,c+1] == copyImage[r+1+mask[h][0], c+1+mask[h][1]]) and (copyImage[r+1,c+1] !=
                copyImage[r+1+mask[h+1][0], c+1+mask[h+1][1]] or copyImage[r+1,c+1] != copyImage[r+1+mask[h+5][0]
                , c+1+mask[h+5][1]]):
                qNum += 1
```

```
# Check if the pixel is edge part(q = 1)
if qNum == 1:
    resultImage[r,c] = 0 # Erase the pixel(i.e. turn it into black pixel)
    copyImage[r+1,c+1] = 0
```

## Result: