Computer Vision HW7

R12922054 資工所 邱信瑋

1.Write a program which does thinning on a downsampled image (lena.bmp).

How to implement :

1. step 1: Binarize the benchmark image lena as in HW2.

```
for i in range(row_size):
    for j in range(col_size):
        np_img[i][j] = 255 if(np_img[i][j] >= 128) else 0
```

2. step 2: Downsampling Lena from 512x512 to 64x64.

```
new_size = 64
new_np_img = np.zeros((new_size, new_size), np.int8)
step_row = row_size // new_size
step_col = col_size // new_size
for i in range(0, row_size, step_row):
    for j in range(0, col_size, step_col):
        new_i = i // step_row
        new_j = j // step_col
        new_np_img[new_i][new_j] = np_img[i][j]
```

3. step 3: Do thinning operator iteratively until it would not change according to the pdf, Thinning Operator, Figure 1 Figure 2.

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

Figure 1: pairOperator_ref.

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

Figure 2: shrinkOperator_ref.

```python
def Yokoi_Operator(np_img):

    row_size = np_img.shape[0]
    col_size = np_img.shape[1]
    Yokoi_array = np.zeros((row_size, col_size), np.int32)

    # x7  x2  x6
    # x3  x0  x1
    # x8  x4  x5
    for i in range (row_size):
        for j in range (col_size):

            x0 = np_img[i][j]
            x1 = 0 if j == col_size - 1 else np_img[i][j + 1]
            x2 = 0 if i == 0 else np_img[i - 1][j]
            x3 = 0 if j == 0 else np_img[i][j - 1]
            x4 = 0 if i == row_size - 1 else np_img[i + 1][j]
            x5 = 0 if (i == row_size - 1 or j == col_size - 1) else np_img[i + 1][j + 1]
            x6 = 0 if (i == 0 or j == col_size - 1) else np_img[i - 1][j + 1]
            x7 = 0 if (i == 0 or j == 0) else np_img[i - 1][j - 1]
            x8 = 0 if (i == row_size - 1 or j == 0) else np_img[i + 1][j - 1]

            a = ''
            if x0 != 0:
                a += h(x0, x1, x6, x2)
                a += h(x0, x2, x7, x3)
                a += h(x0, x3, x8, x4)
                a += h(x0, x4, x5, x1)
                label = f(a)
            else:
                label = 7
            Yokoi_array[i][j] = label
    return Yokoi_array
```

Figure 3: yokoiOperator_code.

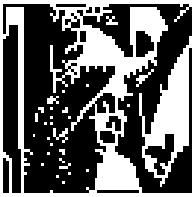
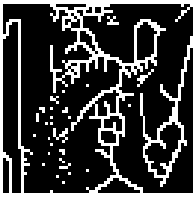
```python
def Pair_Operator(np_img):

    m = 1
    row_size = np_img.shape[0]
    col_size = np_img.shape[1]
    pair_list = list()
    h = (lambda a, m : 1 if a == m else 0)
    for i in range(row_size):
        for j in range(col_size):
            x0 = np_img[i][j]
            if x0 != 7 :
                x1 = 0 if j == col_size - 1 else np_img[i][j + 1]
                x2 = 0 if i == 0 else np_img[i - 1][j]
                x3 = 0 if j == 0 else np_img[i][j - 1]
                x4 = 0 if i == row_size - 1 else np_img[i + 1][j]

                sum_h = h(x1, m) + h(x2, m) + h(x3, m) + h(x4, m)
                pair_list.append('p' if (sum_h >= 1 and x0 == m) else 'q')
            else:
                pair_list.append('g')

    np_pair = np.array(pair_list).reshape((row_size, col_size))
    return np_pair
```

Figure 4: pairOperator_code.

```python
def Shrink_Operator(np_pair):

    row_size = np_pair.shape[0]
    col_size = np_pair.shape[1]
    output = np.zeros((row_size, col_size), np.int32)

    h = (lambda b, c, d, e : 1 if ( (c != 'g') and ( (d == 'g') or ( e == 'g' )) ) else 0)
    f = (lambda a1, a2, a3, a4, x : 'g' if ( a1+a2+a3+a4 ) == 1 else x)

    for i in range(row_size):
        for j in range(col_size):
            x0 = np_pair[i][j]
            if x0 == 'p':

                x1 = 'g' if j == col_size - 1 else np_pair[i][j + 1]
                x2 = 'g' if i == 0 else np_pair[i - 1][j]
                x3 = 'g' if j == 0 else np_pair[i][j - 1]
                x4 = 'g' if i == row_size - 1 else np_pair[i + 1][j]
                x5 = 'g' if (i == row_size - 1 or j == col_size - 1) else np_pair[i + 1][j + 1]
                x6 = 'g' if (i == 0 or j == col_size - 1) else np_pair[i - 1][j + 1]
                x7 = 'g' if (i == 0 or j == 0) else np_pair[i - 1][j - 1]
                x8 = 'g' if (i == row_size - 1 or j == 0) else np_pair[i + 1][j - 1]

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

                np_pair[i][j] = f(a1, a2, a3, a4, x0)

    for i in range(row_size):
        for j in range(col_size):
            if np_pair[i][j] != 'g':
                output[i][j] = 255
    return output
```

Figure 5: shrinkOperatpr_code.

4. step 4: Results.

iterative 1	iterative 2	iterative 3
		
iterative 4	iterative 5	iterative 6
		
iterative 7