Computer Vision Homework 6

Name: 黃新予

Student ID: f08922136

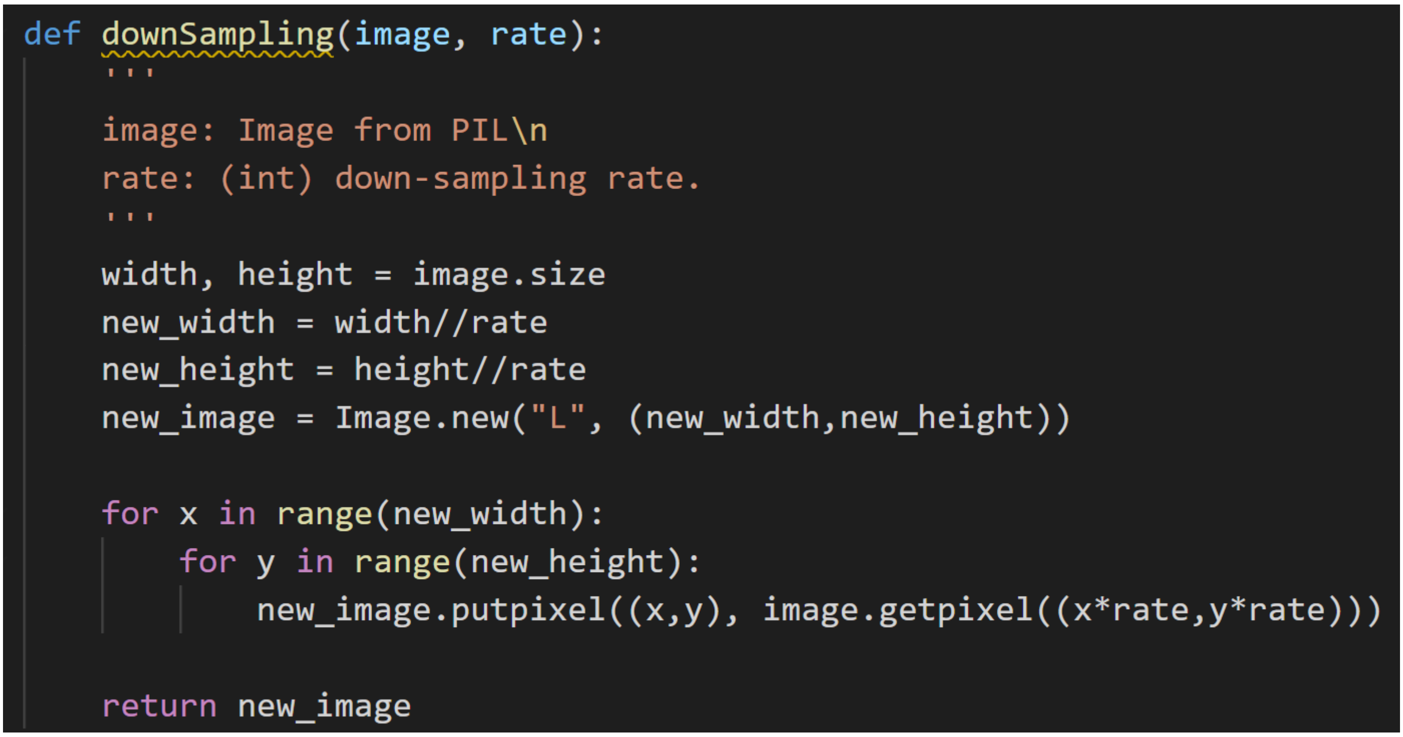
I. **Environment Setup**

Language: Python 3 (on VS code)

Library: numpy, PIL

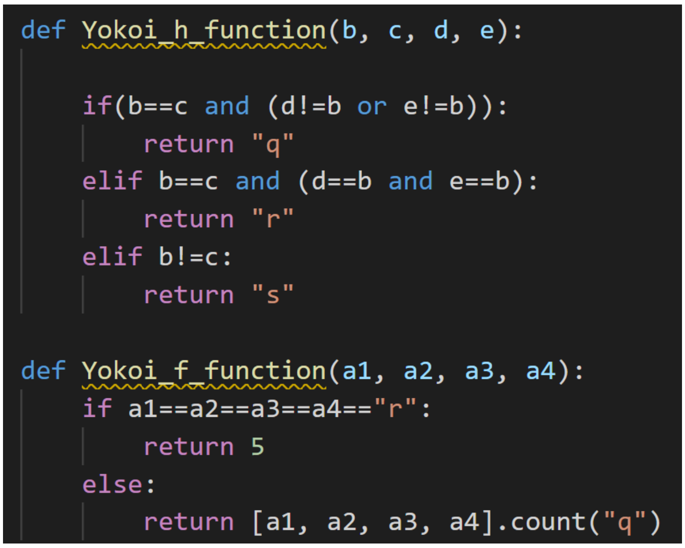
II. Q1: Write a program which counts the Yokoi connectivity number on a downsampled image(lena.bmp).

step 1: down-sampling the original binarized lean.bmp from 512*512 to 64*64



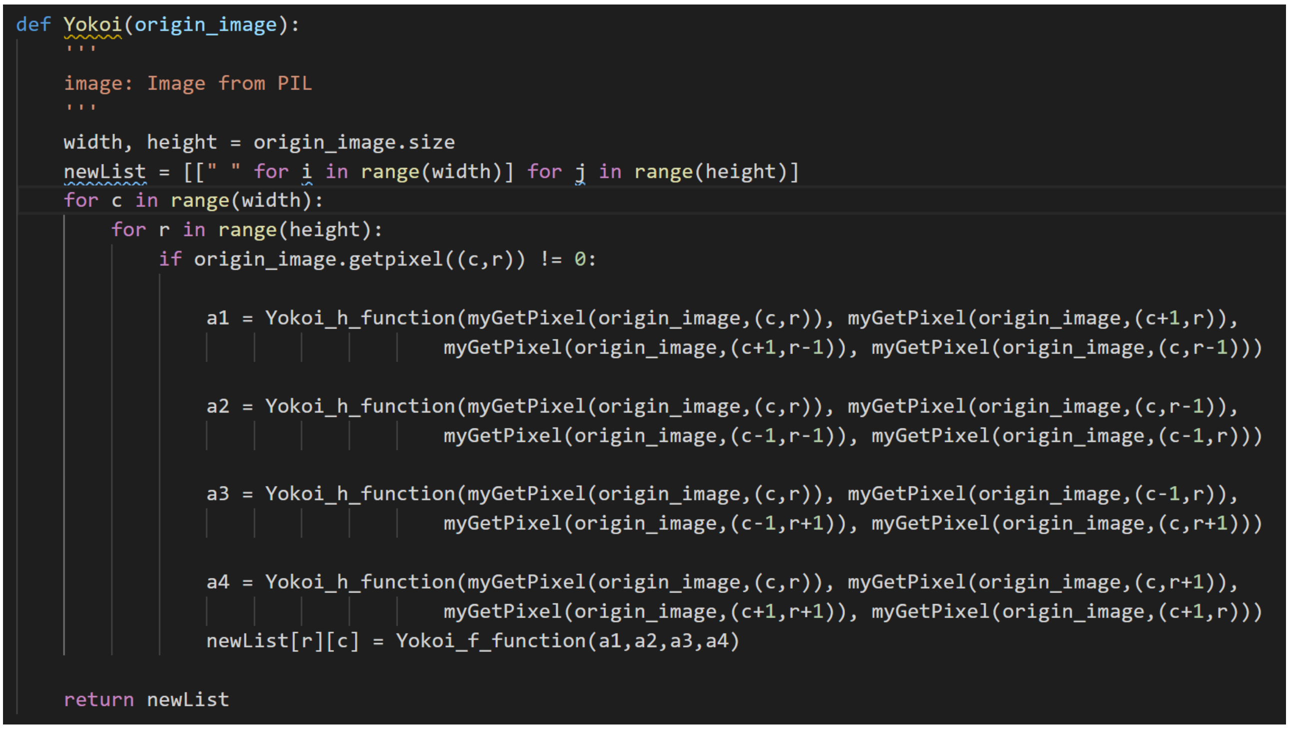
```python
def downSampling(image, rate):
    '''
    image: Image from PIL\n
    rate: (int) down-sampling rate.
    '''
    width, height = image.size
    new_width = width//rate
    new_height = height//rate
    new_image = Image.new("L", (new_width,new_height))

    for x in range(new_width):
        for y in range(new_height):
            new_image.putpixel((x,y), image.getpixel((x*rate,y*rate)))

    return new_image
```

step 2: implement Yokoi h function and f function



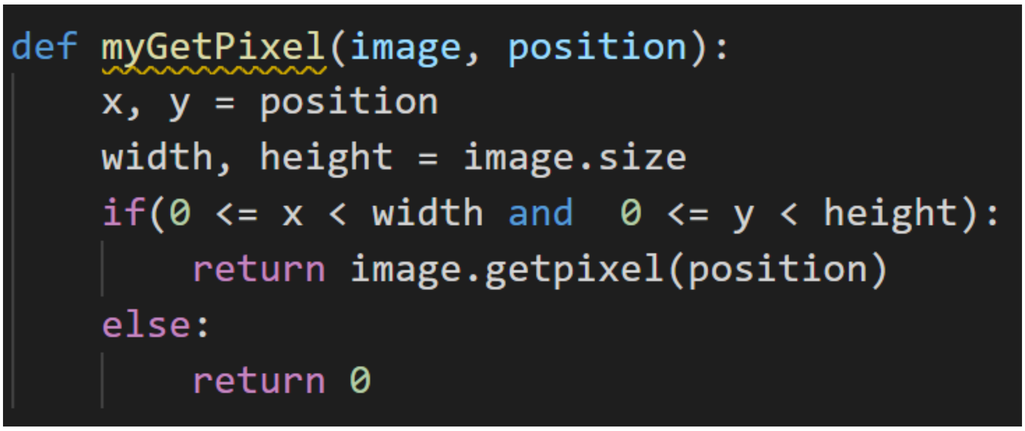
```python
def Yokoi_h_function(b, c, d, e):

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

Step 3: traverse each element in the down-sampling image and count the connectivity number by h function and f function



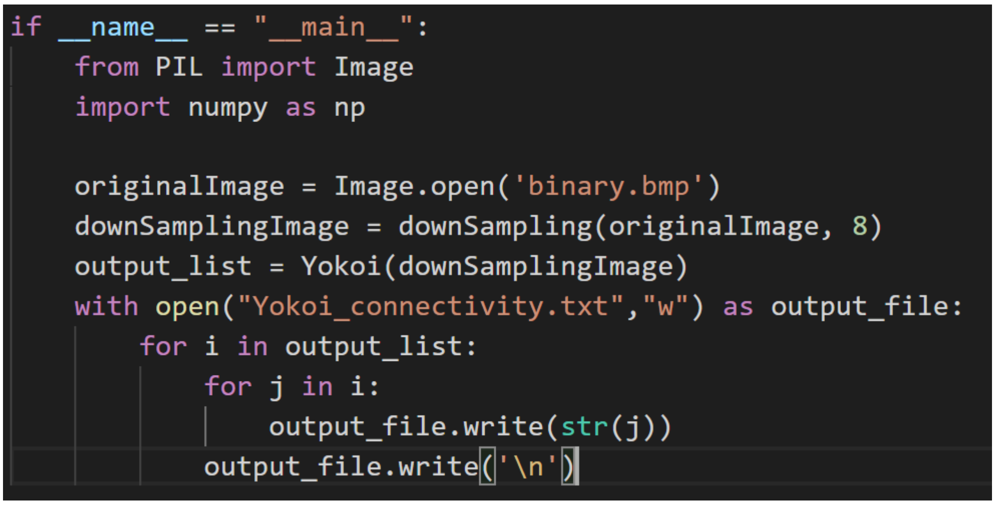
```python
def Yokoi(origin_image):
    '''
    image: Image from PIL
    '''
    width, height = origin_image.size
    newList = [[" " for i in range(width)] for j in range(height)]
    for c in range(width):
        for r in range(height):
            if origin_image.getpixel((c,r)) != 0:

                a1 = Yokoi_h_function(myGetPixel(origin_image,(c,r)), myGetPixel(origin_image,(c+1,r)),
                                    myGetPixel(origin_image,(c+1,r-1)), myGetPixel(origin_image,(c,r-1)))

                a2 = Yokoi_h_function(myGetPixel(origin_image,(c,r)), myGetPixel(origin_image,(c,r-1)),
                                    myGetPixel(origin_image,(c-1,r-1)), myGetPixel(origin_image,(c-1,r)))

                a3 = Yokoi_h_function(myGetPixel(origin_image,(c,r)), myGetPixel(origin_image,(c-1,r)),
                                    myGetPixel(origin_image,(c-1,r+1)), myGetPixel(origin_image,(c,r+1)))

                a4 = Yokoi_h_function(myGetPixel(origin_image,(c,r)), myGetPixel(origin_image,(c,r+1)),
                                    myGetPixel(origin_image,(c+1,r+1)), myGetPixel(origin_image,(c+1,r)))
                newList[r][c] = Yokoi_f_function(a1,a2,a3,a4)

    return newList
```

I also implement myGetPixel function to detect out of range problem.



```python
def myGetPixel(image, position):
    x, y = position
    width, height = image.size
    if(0 <= x < width and  0 <= y < height):
        return image.getpixel(position)
    else:
        return 0
```

Step 5: write the output list to a txt file



```python
if __name__ == "__main__":
    from PIL import Image
    import numpy as np

    originalImage = Image.open('binary.bmp')
    downSamplingImage = downSampling(originalImage, 8)
    output_list = Yokoi(downSamplingImage)
    with open("Yokoi_connectivity.txt","w") as output_file:
        for i in output_list:
            for j in i:
                output_file.write(str(j))
            output_file.write('\n')
```

Result: 0s are also recorded.

```
11111111        12111111111122322221     111111111111       0  0
15555551         115555555511 2 11  11   1155555555511        0
15555551        1 2115555112  21112221    155555555551      21
15555551        1 2 155112 22221511       1555555555511     1
15555551         22 2112 22    121 0 0    15555555555511   0
15555551         1  2  21 2     1   1     15555555555551  0
15555551           12 1  121111    1321   155555555555511
15111551           1322 1155551111        155555555555551
111 1551            1  121555555511       155555555555511
11  1551                 21155555511      15511155555511
21  1551                 2 15555555111    1551 11555511
1   1551                 2 155555555511   1551  115551         1
    1551               1121155555555551   1551   15511        12
    1551               15555555555555511  1551   1111        111
    1551        1     2221155555555555511 1151    11        1151
    1551        2    22 1 1555555555555511 151  11111       1551
    1551        2    1   11555555555555551 151 115551      11551
    1551        2       11555555555555555111511155511     115551
    1551       12      1155555555555555555555555551      155551
    1551       11   0 221555555555555555555555555112     1155551
    1551       111   22 15555555555555555555555551 1     1555551
    1551       1511  1 125112111121115555555555111      11555551
    1551       15521  1 121 1 11  1  15555555111  0     15555551
    1551       1151  132 2          1155555111   0     115555551
    1551        151 0  322         115555111  121      155555551
    1551        1221   2           1555551   131      1155555551
    1551         2  0  1          115555511   1       1155555551
    1551         2   0      0    1155555551  0       1 155555551
    1551         2              11555555551          21155555551
    1551         1  0          115555555551          15555555551
    1551          1           11511115555521  1     115555555551
    1551        1 1          11111  1155511   2     155555555551
    1551       131           111     15111    2     155555555551
    1551      121 0        1121   1  111  1   2    1155555555551
    1551      11           111 1  221 11  1   2    1555555555551
    1551    12  0    1     21 121  11 1111    2    1555555555551
    1551     1      12    22  151111111551    2   11555555555551
    1551   1              2   1555551115511   1   15555555555551
    1551   2    0     0  22  12555551 15551    1  15555555555551
    1551   1             1    1555511 11511    2 115555555555551
    1551      0 0      21     155551 1 151     2 155555555555551
    1551               2      1555112  151     2 155555555555551
    1551         1   1 1     1155555551111     2 155555555555551
    1551         2  22       111511111212      21155555555555551
    1551  0      1 12          151    2 1      15555555111555551
    1551       0  0  0         1111  121       155555551 1555551
    1551            0           11111111       155555551 1555551
    1551        0                115551        155555551 1555511
    1551                          15551        211111111 155511
    11521      1   12          122155511       2     11 115511
1    151 0     1    1            155555111     2111     15511
22   1511          1             15555555111   155111   1511
 22  1511          1             15555555551   155551  1151
  2  151            0 1        11155555555511  155511  1511
  2  1521    0        1        155555555555511 15551 12151
  2  151           121         155555555555551 155511 1551
  2  1511                   0  155555555555551 115551 1511
  21 1511            11        155555555555551  111111151
  11 151          0           11555555555555511    111511
  11 151                      15555555555555551      151
  11 151            0        115555555555555551      211
  11 151                     1155555555555555511     1
  11 151                    0 155555555555555551
  11 111        0            1211111111111111111
```