Computer Vision Homework 7

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 does thinning on a downsampled image (lena.bmp).
step 1: binarize the lena.bmp
step 2: down-sampling the original binarized lean.bmp from 512*512 to 64*64

```
if __name__ == "__main__":
    origin_image = Image.open('lena.bmp')
    binary_image = binary(origin_image,128)
    down_sample_image = down_sample(binary_image, 8)
```

step 3: according to the announcement, implement the thinning operator.
step 3-1: create Yokoi connectivity number array (use the function from homework 6)

step 3-2: use Yokoi connectivity number created from step 3-1 to create Pair array
a). implement **Pair Relationship Operator h function**

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

b) implement **Pair Relationship Operator output function**

```
def pair_operator_output(x0, m, Yokoi_Array):
    counter = 0
    neighborPixels = neighbor_pixel(x0, Yokoi_Array)

    for i in neighborPixels:
        counter += pair_operator_h_function(i, m)
    if counter < 1 or Yokoi_Array[x0] != m:
        return "q"
    else:
        return "p"
```

c) implement **Pair Relationship Operator**

```
def pair_operator(image, Yokoi_Array):
    Pair_Array = np.full(image.size, ' ')
    width, height = image.size
    for r in range(height):
        for c in range(width):
            if Yokoi_Array[r, c] != ' ':
                Pair_Array[r, c] = pair_operator_output((r,c), '1', Yokoi_Array)
    return Pair_Array
```

step 3-3: using Pair array created from step 3 to implement **Connected Shrink Operator**

a) implement **Connected_Shrink_Operator h function**

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

b) implement **Connected_Shrink_Operator f function**

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

c) using pair array and cso h function and cso function to update the original image

```
def Connected_Shrink_Operator(origin_image, Pair_Array):
    width, height = origin_image.size
    for r in range(height):
        for c in range(width):
            if Pair_Array[r, c] == 'p':
                a1 = CSO_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 = CSO_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 = CSO_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 = CSO_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)))
                origin_image.putpixel((c,r), CSO_f_function(a1, a2, a3, a4, origin_image.getpixel((c,r))))

    return origin_image
```

step 4: repeat step 3 for 7 time and get the result image

```
for i in range(7):
    # thinning start
    Yokoi_Array = Yokoi(down_sample_image)
    Pair_Array = pair_operator(down_sample_image, Yokoi_Array)
    down_sample_image = Connected_Shrink_Operator(down_sample_image, Pair_Array)
down_sample_image: Image
down_sample_image.show()
```

result image: