d11922022 資工所 楊寓鈞

Environment

- python3.9
- Pillow 10.0.0, numpy 1.25.2, pandas 2.0.3

basic setups and utility functions

```
from PIL import Image
import numpy as np
import copy

img = Image.open('./lena.bmp') # load lena.bmp
img_array = np.array(img) # pixel content saved in np.array
width, height = img_array.shape # get `width` and `height`
img_list = img_array.tolist() # transform pixel content into list

def save_image(img, path='./lena.bmp'):
        img_ = Image.fromarray(np.array(img, dtype='uint8'), mode='L')
        img_.save(path)
        return img_

def binarize(img, height=height, width=width):
        for y in range(height):
                for x in range(width):
                        img[y][x] = 255 if img[y][x] >= 128 else 0
        return img

def shrink(img, height=height, width=width, scale=2):
        for y in range(0, height, 2):
                for x in range(0, width, 2):
                        elm = img[y][x]
                        img[y//scale][x//scale] = elm
                        img = [ [img[y][x] for x in range(0, width//scale)]
for y in range(0, height//scale)]
        return img
```

```
result = copy.deepcopy(img_list)
result = binarize(result)
result = shrink(result, scale=8)
```

```
width, height = width//8, height//8
save_image(result, './binarized_shrink_8.bmp')
```

now `result` contains down-sampled by 8 image.

Yokoi connectivity

```
# in the manner of [row, col]
k1 = [(0,1),(-1,1),(-1,0)]
k2 = [(-1,0),(-1,-1),(0,-1)]
k3 = [(0,-1),(1,-1),(1,0)]
k4 = [(1,0),(1,1),(0,1)]
```

```
def matrix2text(matrix, file='yokoi_matrix.txt', height=height,
width=width):
        with open(file, 'w') as f:
                for y in range(height):
                        for x in range(width):
                                s = str(matrix[y][x]) if matrix[y][x] else '
'
                                f.write(s)

                        f.write('\n')

def h(b, c, d, e):
        """
        b: center
        c: 1st pixel in the kernel, connected with b
        d: 2nd pixel in the kernel
        e: 3rd pixel in the kernel
        """
        bc, bd, be = (b == c, b == d, b==e)
        if not bc:
                return 's'
        elif bd and be:
                return 'r'
        else:
                return 'q'

def f(a1, a2, a3, a4):
        cnt = {'s':0,'q':0,'r':0}
        cnt[a1] = cnt[a1]+1
        cnt[a2] = cnt[a2]+1
        cnt[a3] = cnt[a3]+1
        cnt[a4] = cnt[a4]+1
```

```
        if cnt['r'] == 4:
                return 5
        else:
                return cnt['q']

def get_kernel_pixels(img, y, x, kernel, height=height, width=width):
        return [img[y+y_][x+x_] if 0 <= y+y_ < height and 0 <= x+x_ < width
else 0 for y_, x_ in kernel ]

def yokoi(img, kernels, height=height, width=width):
        matrix = [ [0 for x in range(width)] for y in range(height)]
        for y in range(height):
                for x in range(width):
                        if img[y][x]:
                                a = []
                                for kernel in kernels:
                                        a.append(
                                                h(img[y][x],
*get_kernel_pixels(img, y, x, kernel, height, width))
                                        )

                                matrix[y][x] = f(*a)
        return matrix

matrix = yokoi(result, [k1, k2, k3, k4])
matrix2text(matrix)
```

1. function `h(b, c, d, e)` as the definition in the lecture slide
2. function `f(a1, a2, a3, a4)` as the definition in the lecture slide, but use `dict` to do efficient counts of labels
3. function `get_kernel_pixels` apply corner kernel to extract 3 corner pixel values, when kernel is out side of the boundary, just give 0 as it's value
4. traverse over every pixel to get the right yokoi connectivity number

the result is in `yokoi_matrix.txt`