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_
```

```
# in the manner of [row, col]
# kernel = [[0,1,1,1,0],
#           [1,1,1,1,1],
#           [1,1,1,1,1],
#           [1,1,1,1,1],
#           [0,1,1,1,0]]
kernel = [(-2,-1),(-2,0),(-2,1),
                  (-1,-2),(-1,-1),(-1,0),(-1,1),(-1,2),
                  (0,-2),(0,-1),(0,0),(0,1),(0,2),
                  (1,-2),(1,-1),(1,0),(1,1),(1,2),
                  (2,-1),(2,0),(2,1),]
```

a. dilation

```
def dilation(img, kernel=kernel, height=height, width=width, threshold=255):
        dilation = [ [0 for x in range(width)] for y in range(height)]
        for y in range(height):
                for x in range(width):
                        for dy, dx in kernel:
```

```
                                y_, x_ = y+dy, x+dx
                                if 0<= y_ < height and 0 <= x_ < width:
                                        dilation[y][x] = max(img[y_][x_],
dilation[y][x])
        return dilation

result = copy.deepcopy(img_list)
save_image(dilation(result, kernel), './dilation.bmp')
```

1. the function traverses through all pixels
2. take each pixel as the center pixel and set the maximum value of all kernel applied pixel as center pixel value

b. erosion

```
def erosion(img, kernel=kernel, height=height, width=width):
        erosion = [ [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] == 255:
```

```
                                for dy, dx in kernel:
                                        y_, x_ = y+dy, x+dx
                                        if not (0<= y_ < height and 0 <= x_
< width and img[y_][x_] == 255):
                                                break

                                else:
                                        erosion[y][x] = 255

        return erosion

result = copy.deepcopy(img_list)
result = binarize(result)
save_image(erosion(result, kernel), './erosion.bmp')
```

1. the function traverses through all pixels
2. take each pixel as the center pixel and set the minimum value of all kernel applied pixel as center pixel value

c. opening

```
def opening(img, kernel=kernel, height=height, width=width):
        return dilation(erosion(img, kernel), kernel)

result = copy.deepcopy(img_list)
result=binarize(result)
save_image(opening(result, kernel), './opening.bmp')
```

1. the function applies erosion first then dilation next with the kernel

d. closing

```
def closing(img, kernel=kernel, height=height, width=width):
        return erosion(dilation(img, kernel), kernel)

result = copy.deepcopy(img_list)
result=binarize(result)
save_image(closing(result, kernel), './closing.bmp')
```

1. the function applies dilation first then erosion next with the kernel