

HW 5 Mathematical Morphology

Mathematical Morphology - Gray Scale Morphology

Source code

Please refer to the file "hw5.py" within the same folder as this report document.

There are two main functions for this homework:

1. dilation(src, kernel)

```
48 """
49     This function does the dilation operation.
50     @param src should be an image
51     @param kernel should be an instance of Kernel class
52     @param threshold should be an integer (default 128 if not given)
53     @return a resulted image
54 """
55 def dilation(src, kernel):
56     newImage = Image.new(src.mode, src.size)
57     newImagePixels = newImage.load()
58     for i in range(src.size[0]):
59         for j in range(src.size[1]):
60             localMax = 0
61             for direction in kernel.get_directions():
62                 new_i = i + direction[0]
63                 new_j = j + direction[1]
64                 if new_i >= 0 and new_i < src.size[0] and new_j >= 0 and new_j < src.size[1]:
65                     localMax = max(localMax, src.getpixel((new_i, new_j)))
66             newImagePixels[i, j] = localMax
67     return newImage
```

2. erosion(src, kernel)

```
69 """
70     This function does the erosion operation.
71     @param src should be an image
72     @param kernel should be an instance of Kernel class
73     @param threshold should be an integer (default 128 if not given)
74     @return a resulted image
75 """
76 def erosion(src, kernel):
77     newImage = Image.new(src.mode, src.size)
78     newImagePixels = newImage.load()
79     for i in range(src.size[0]):
80         for j in range(src.size[1]):
81             set = True
82             localMin = 255
83             for direction in kernel.get_directions():
84                 new_i = i + direction[0]
85                 new_j = j + direction[1]
86                 if new_i >= 0 and new_i < src.size[0] and new_j >= 0 and new_j < src.size[1]:
87                     localMin = min(localMin, src.getpixel((new_i, new_j)))
88                 continue
89             else:
90                 set = False
91                 break
92             if set:
93                 newImagePixels[i, j] = localMin
94     return newImage
```

Kernel Representation

A class is defined for kernel representation:

```
10 class Kernel:
11     def __init__(self, init_list, origin):
12         self.pattern = init_list
13         self.origin = origin
14
15     """
16     @return a list of directions, i.e., list of 2-tuples
17     """
18     def get_directions(self):
19         tmp_list = []
20         for i in range(len(self.pattern)):
21             for j in range(len(self.pattern[0])):
22                 if self.pattern[i][j] != "x":
23                     direction = (j - self.origin[0], i - self.origin[1])
24                     tmp_list.append(direction)
25         return tmp_list
```

get_directions(self)

it returns list of directions, i.e., “vectors”, for us to traverse the kernel.

The kernel patterns used in this homework

```
31 octo_kernel_pattern = [
32     ["x", 0, 0, 0, "x"],
33     [0, 0, 0, 0, 0],
34     [0, 0, 0, 0, 0],
35     [0, 0, 0, 0, 0],
36     ["x", 0, 0, 0, "x"]
37 ]
```

← We use `oct_kernel_pattern` with `origin = (2, 2)`

Result

(the resulted images are saved properly within the same folder as well)



↑ Dilation.bmp



↑ Erosion.bmp



↑ Opening.bmp



↑ Closing.bmp