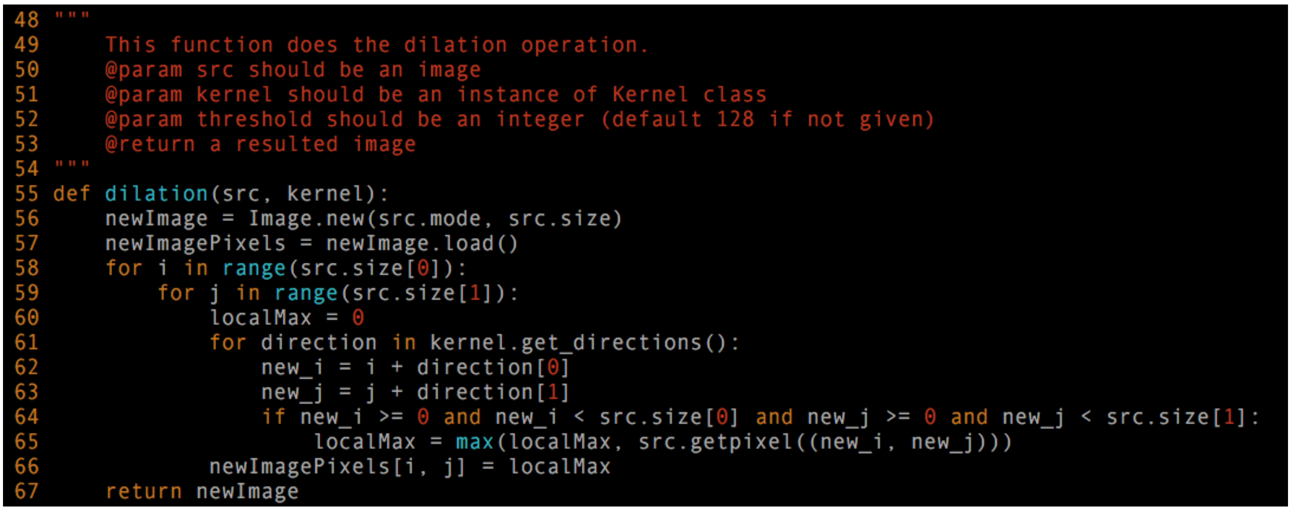
HW 5 Mathematical Morphology

Mathematical Morphology - Gray Scale Morphology

Source code

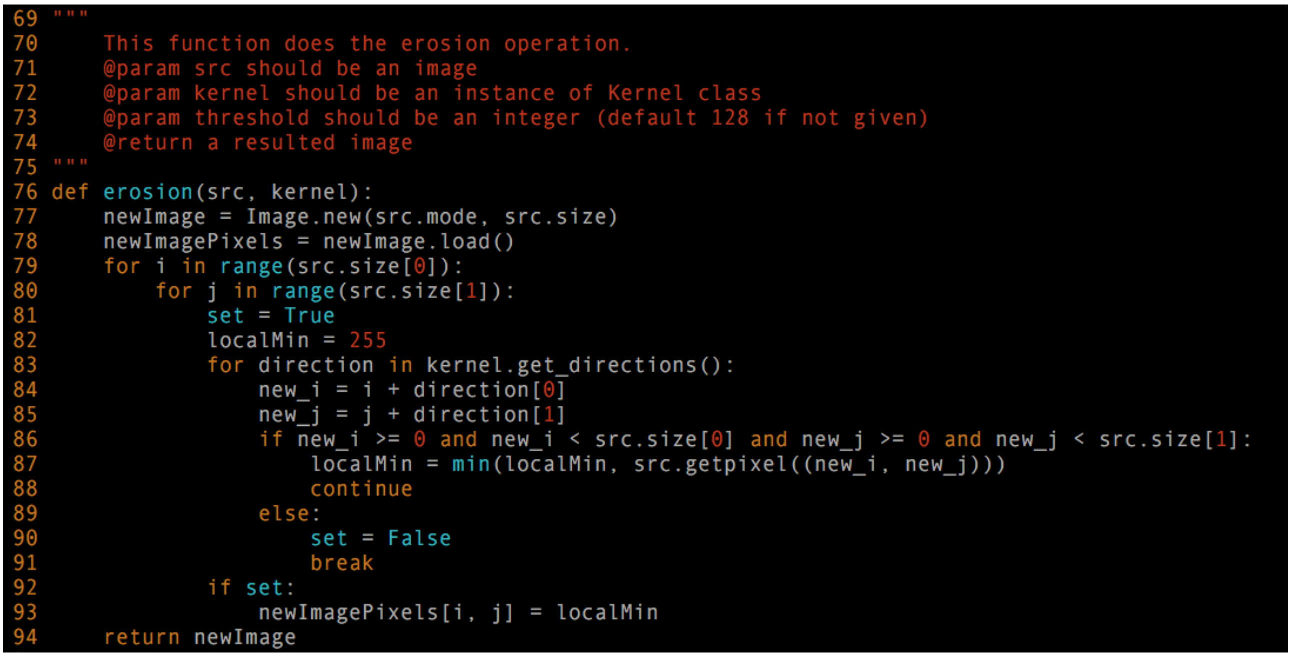
Please refer to the file “hw5.py” within the same folder as this report document.
There are two main functions for this homework:

1. dilation(src, kernel)



```
"""
    This function does the dilation operation.
    @param src should be an image
    @param kernel should be an instance of Kernel class
    @param threshold should be an integer (default 128 if not given)
    @return a resulted image
"""
def dilation(src, kernel):
    newImage = Image.new(src.mode, src.size)
    newImagePixels = newImage.load()
    for i in range(src.size[0]):
        for j in range(src.size[1]):
            localMax = 0
            for direction in kernel.get_directions():
                new_i = i + direction[0]
                new_j = j + direction[1]
                if new_i >= 0 and new_i < src.size[0] and new_j >= 0 and new_j < src.size[1]:
                    localMax = max(localMax, src.getpixel((new_i, new_j)))
            newImagePixels[i, j] = localMax
    return newImage
```

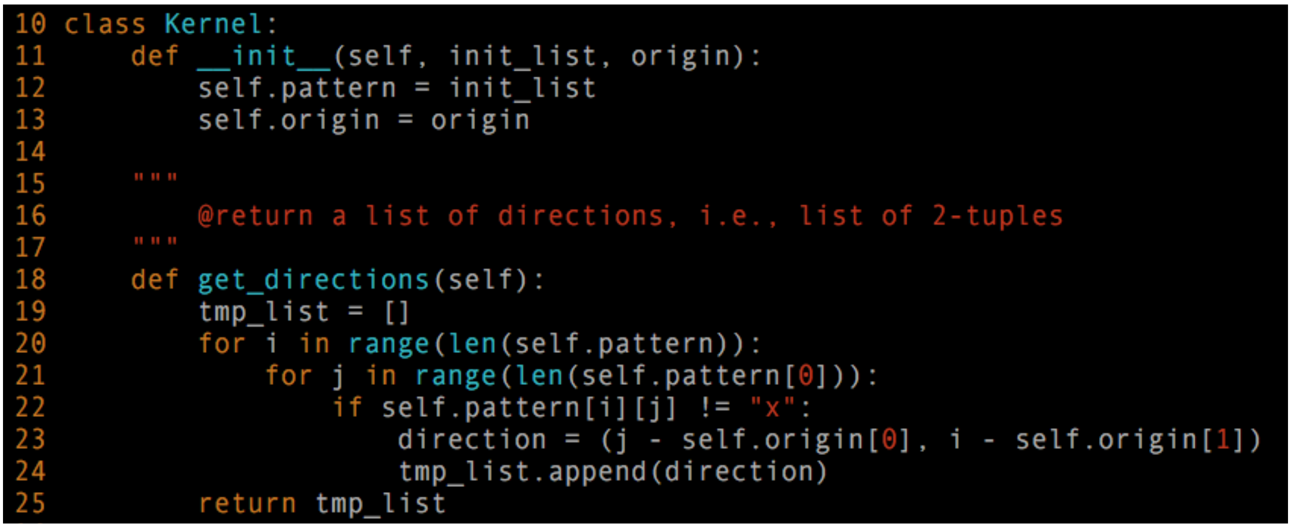
2. erosion(src, kernel)



```
"""
    This function does the erosion operation.
    @param src should be an image
    @param kernel should be an instance of Kernel class
    @param threshold should be an integer (default 128 if not given)
    @return a resulted image
"""
def erosion(src, kernel):
    newImage = Image.new(src.mode, src.size)
    newImagePixels = newImage.load()
    for i in range(src.size[0]):
        for j in range(src.size[1]):
            set = True
            localMin = 255
            for direction in kernel.get_directions():
                new_i = i + direction[0]
                new_j = j + direction[1]
                if new_i >= 0 and new_i < src.size[0] and new_j >= 0 and new_j < src.size[1]:
                    localMin = min(localMin, src.getpixel((new_i, new_j)))
                    continue
                else:
                    set = False
                    break
            if set:
                newImagePixels[i, j] = localMin
    return newImage
```

Kernel Representation

A class is defined for kernel representation:



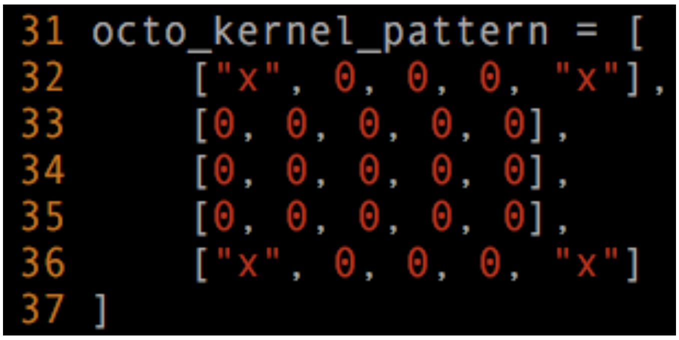
```python
class Kernel:
    def __init__(self, init_list, origin):
        self.pattern = init_list
        self.origin = origin

    """
        @return a list of directions, i.e., list of 2-tuples
    """
    def get_directions(self):
        tmp_list = []
        for i in range(len(self.pattern)):
            for j in range(len(self.pattern[0])):
                if self.pattern[i][j] != "x":
                    direction = (j - self.origin[0], i - self.origin[1])
                    tmp_list.append(direction)
        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

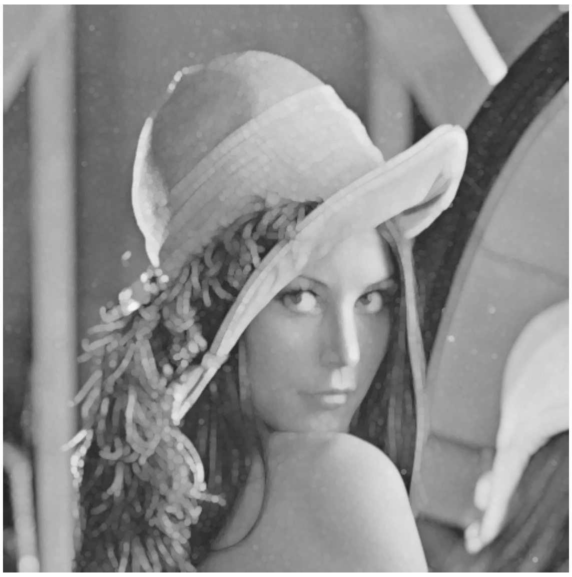


```python
octo_kernel_pattern = [
        ["x", 0, 0, 0, "x"],
        [0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0],
        ["x", 0, 0, 0, "x"]
]
```

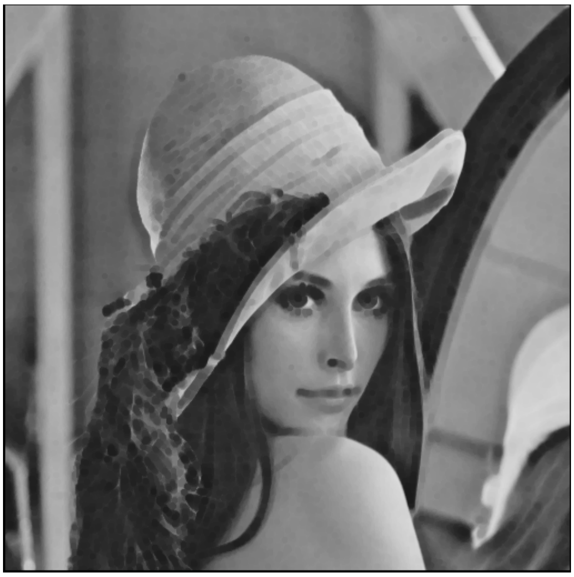
← We use `oct_kernel_pattern` with origin = (2, 2)

Result

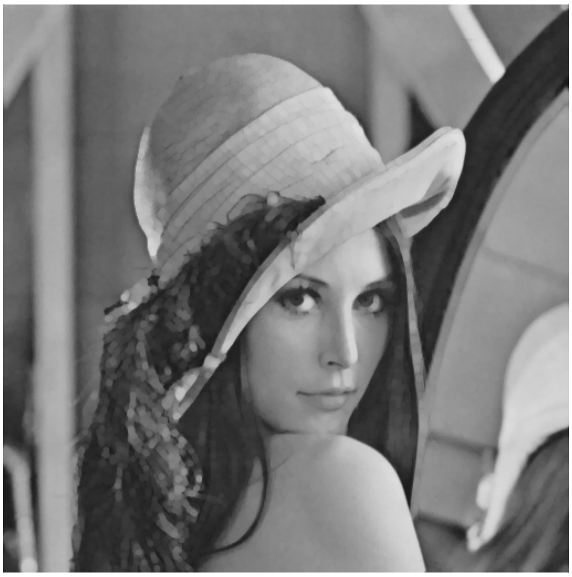
(the resulted images are saved properly within the same folder as well)



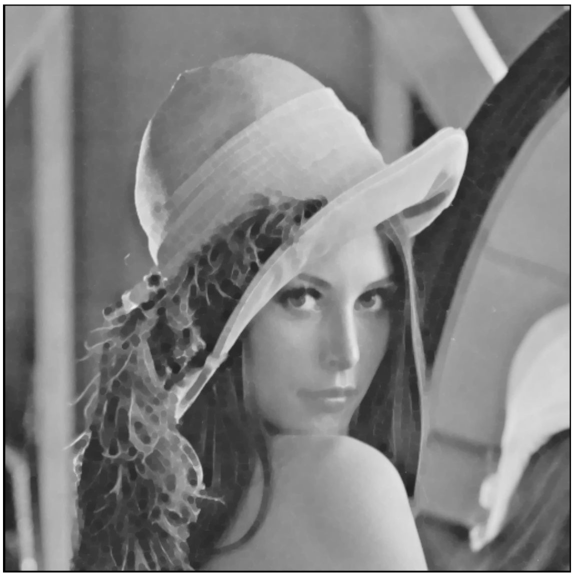
↑ Dilation.bmp



↑ Erosion.bmp



↑ Opening.bmp



↑ Closing.bmp